**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО»**
**Факультет інформатики та обчислювальної техніки**
**Кафедра автоматики та управління в технічних системах**

«До захисту допущено»
Завідувач кафедри
_____ О.І. Ролік
«___»_____2019 р.

# Дипломний проект

**на здобуття ступеня бакалавра**

**з напряму підготовки 6.050103 «Програмна інженерія»**

**на тему: «Система тестування для навчальних закладів»**

Виконав (-ла):
студент (-ка) IV курсу, групи ІТ-51
Літвінова Наталія Олександрівна

_____

Керівник:
Ст.викладач Тимофєєва Ю.С.

_____

Рецензент:
Директор КМП «Інформпроект» Порутенко В.Г.

_____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2019 рік

**Національний технічний університет України**
**«Київський політехнічний інститут імені Ігоря Сікорського»**
**Факультет інформатики та обчислювальної техніки**
**Кафедра автоматики та управління в технічних системах**

Рівень вищої освіти – перший (бакалаврський)
Напрям підготовки – 6.050103 «Програмна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.І. Ролік
«___»_____2019 р.

**ЗАВДАННЯ**
**на дипломний проект студенту**
**Літвіновій Наталії Олександрівні**

1.Тема проекту «Система тестування для навчальних закладів», керівник проекту ст. викладач Тимофєєва Юлія Сергіївна, затверджені наказом по університету від «___»_____ 2019 р. №_____

2. Термін подання студентом проекту _____

3. Вихідні дані до проекту

Операційна система Windows 10, мова програмування C#, середовище програмування Visual Studio 2019, цільова платформа .Net Framework 4, обрана для розробки технологія –WPF, СУБД – MySQL

4. Зміст пояснювальної записки

1. Вступ 2. Перелік скорочень, умовних позначень, термінів 3. Постановка задачі 4. Вибір та обгрунтування компонентів системи 5. Архітектура проекту 6. Реалізація моделі даних 7. Опис програми 8. Сценарії використання програми

***Додатки:*** 9. Приклад XML-пакету тесту 10. Приклад збереженої процедури 11. Код програми

5. Перелік графічного матеріалу

Діаграма прецедентів, діаграма класів, ER-діаграма, діаграма діяльності

6. Дата видачі завдання _____

Календарний план

| № з/п | Назва етапів виконання дипломного проекту | Термін виконання етапів проекту | Примітка |
|---|---|---|---|
| 1 | Вибір тематичного напрямку та узгодження теми курсової роботи | 22.02.2019 | |
| 2 | Аналіз теоретичних матеріалів та вивчення предметної області | 15.04.2019 | |
| 3 | Розробка технічного завдання, вибір методів та засобів реалізації задачі | 24.04.2019 | |
| 4 | Огляд існуючих рішень з тематики роботи | 27.04.2019 | |
| 5 | Розробка структури БД та проектування системи | 06.05.2019 | |
| 6 | Реалізація проекту | 20.05.2019 | |
| 7 | Налагодження та перевірка програми | 23.05.2019 | |
| 8 | Оформлення пояснювальної записки | 03.06.2019 | |
| 9 | Передзахист дипломного проекту | 04.06.2019 | |
| 10 | Доопрацювання пояснювальної записки та підготовка презентації | 17.06.2019 | |
| 11 | Захист дипломного проекту | 18.06.2019 | |

Студент Н.О. Літвінова

Керівник проекту Ю.С. Тимофєєва

# АНОТАЦІЯ

Літвінова Н.О. Система тестування для навчальних закладів. КПІ ім. Ігоря Сікорського, Київ, 2019.

Ключові слова: система тестування, електронні тести, навчання, оцінка знань.

Основна частина документу викладена у пояснювальній записці, виконаній на 65 сторінках, та містить 23 рисунки та 17 таблиць. Також до його змісту входить 3 додатки.

Дипломний проект присвячений розробці системи тестування, яка надає можливість проведення оцінки знань та підготовки до контролів у вигляді тестів протягом учбового процесу для навчальних закладів різного рівня.

Програмне забезпечення написане мовою програмування високого рівня C# з використанням об’єктно-орієнтованої парадигми та реалізоване за допомогою WPF та шаблону проектування MVVM. В якості системи управління базами даних обрано MySQL.



# SUMMARY

Litvinova N.O. Testing system for educational institutions. Igor Sikorsky KPI, Kyiv, 2019.

Keywords: testing system, electronic tests, study, knowledge assessment.

The bulk of the document is set out in the explanatory note, with 65 pages, and contains 23 of drawings and 17 tables. Its content also includes 3 applications.

The diploma project is devoted to the development of a testing system, which provides an opportunity to conduct knowledge assessment and preparation for tests in the form of tests during the educational process for educational institutions of different levels.

The software is written in a high level C # programming language using an object-oriented paradigm and implemented with the WPF and the MVVM design template. As a database management system MySQL has been chosen.

| Номер рядка | Формат | Позначення | Найменування | Кіл. листів | № екз. | Примітка |
|---|---|---|---|---|---|---|
| 1 | | | Документація загальна | | | |
| 2 | | | | | | |
| 3 | | | Знову розроблена | | | |
| 4 | | | | | | |
| 5 | А4 | ІТ51.120БАК.002 ПЗ | Пояснювальна записка | | | |
| 6 | | | | | | |
| 7 | А3 | ІТ51.120БАК.003 Д1 | Діаграма прецедентів | 1 | | |
| 8 | | | | | | |
| 9 | А3 | ІТ51.120БАК.004 Д2 | Діаграма класів | 1 | | |
| 10 | | | | | | |
| 11 | А3 | ІТ51.120БАК.005 Д3 | ER-діаграма | 1 | | |
| 12 | | | | | | |
| 13 | А3 | ІТ51.120БАК.006 Д4 | Діаграма діяльності | 1 | | |
| 14 | | | | | | |
| 15 | | | | | | |
| 16 | | | | | | |
| 17 | | | | | | |
| 18 | | | | | | |
| 19 | | | | | | |
| 20 | | | | | | |
| 21 | | | | | | |
| 22 | | | | | | |
| 23 | | | | | | |
| 24 | | | | | | |
| 25 | | | | | | |

| | | | | | ІТ51.12.0БАК.001 ТП | | |
|---|---|---|---|---|---|---|---|
| Змн. | Арк. | № докум. | Підпис | Дата | | | |
| Розроб. | | Літвінова Н.О. | | | Система тестування для навчальних закладів Відомість технічного проекту | Літ. | Арк. 1 / Акрушів 1 |
| Перевір. | | Тимофєєва Ю.С. | | | | | |
| Реценз. | | Порутенко В.Г. | | | | КПІ ім. Ігоря Сікорського ФІОТ, гр. ІТ-51 | |
| Н. Контр. | | Шинкевич М.К. | | | | | |
| Затверд. | | | | | | | |

# Пояснювальна записка
## до дипломного проекту
## на тему: «Система тестування для навчальних закладів»

Київ – 2019 рік

ЗМІСТ

# ВСТУП

Протягом усього свого існування людство прагнуло здобувати нові знання та поліпшувати свої навички, а частина з нас навіть вбачає в цьому сенс свого життя. В цивілізованих країнах наразі неможливо уявити жителя-робітника, який би не мав мінімального рівня освіти.

Останні століття головним способом отримання освіти слугують різноманітні навчальні заклади. Під час учбового процесу, перед співробітниками таких організацій постійно постає питання оцінки набутих знань та неперервного покращення та контролю якості навчання. Тому, зважаючи на актуальність цього питання та тенденцію до комп'ютеризації у сучасному світі, корисним рішенням є створення застосунку, який би допомагав його вирішити.

Даний дипломний проект має на меті створення саме такого настільного застосунку — системи тестування — який можна в загальному випадку класифікувати, як навчальну програму. Система призначена для використання навчальними закладами різного рівня та організаційної структури (курси, школи, університети тощо), а її цільову аудиторію складають викладачі та особи, що навчаються у таких закладах.

Основними задачами, які вирішує дана система, є наступні:

— підготовка вихованців навчальних закладів до запланованих учбовою програмою тестів за допомогою тренувальних тестів;
— проведення контрольних тестів, як одного зі способів оцінки поточних знань;
— створення тестів з різною кількістю відповідей та балами за питання, з можливістю обмеження проходження по даті та часу, а також з налаштуванням доступу тесту лише для певних навчальних груп;
— автоматична оцінка результатів проходження тесту;
— перегляд успішності складання тестів та статистики по тренувальним

тестам.

Розроблене програмне рішення, зокрема, також передбачає можливість локально зберігати файли з тестами у зашифрованому вигляді для тренування в режимі офлайн перед відповідальними тестами.

Сама система спроектована для ОС Windows та для роботи потребує версію .NET Framework не нижче 4. Програма написана на C#, а взаємодія з користувачем відбувається за допомогою WPF, що передбачає реалізацію за допомогою шаблону проектування Model-View-ViewModel.

Пояснювальна записка міститься на 67 сторінках та включає 23 рисунки та 17 таблиць. Проект складається з наступних розділів: вступ, основні розділи, висновки, список використаних джерел та додатки.

# ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

ОС — операційна система

СКБД — система керування базами даних

БД — база даних

Конектор — спосіб взаємодії елементів програмної системи, посилання між двома структурами даних

ODBC (англ. Open Database Connectivity) —програмний інтерфейс доступу до баз даних одночасно для багатьох ОС

MVVM — шаблон проектування Model-View-ViewModel

MVC — шаблон проектування Model-View-Controller

MVP — шаблон проектування Model-View-Presenter

SQL — декларативна мова програмування, що використовується для створення, модифікації та управління даними в реляційній БД, керованої відповідною СКБД

Криптографічна геш-функція — це геш-функція, що приймає довільний блок даних і повертає геш. Вона працює за алгоритмом, при якому будь-яка зміна вхідних даних обов’язково змінить геш-значення

Геш — рядок встановленого розміру, отриманий внаслідок роботи геш-функції

AES — (англ. Advanced Encryption Standard) —симетричний алгоритм блочного шифрування з розміром блока 128 біт та ключем 128/192/256 біт

ЛКМ — ліва кнопка миші

id — ідентифікатор, зазвичай назва унікального поля запису в таблиці БД

ADO — (англ. ActiveX Data Objects) — прикладний програмний інтерфейс для доступу до даних

DLL (англ. Dynamic Link Library) — це бібліотека, яка містить код і дані, що можуть використовуватися кількома програмами одночасно

# 1 ПОСТАНОВКА ЗАДАЧІ

## 1.1 ФУНКЦІОНАЛЬНІ ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Функціональні вимоги — це такі вимоги до програмного забезпечення, які описують очікувану поведінку системи та її функціональні можливості. Наприклад, дії, які може виконати користувач, організацію управління даними, їх обробку та інші специфічні функції, які має виконувати програмний продукт [1].

Оскільки даний проект присвячений розробці системи тестування для перевірки знань під час навчального процесу, то основна частина функціональних вимог регламентує саме функціонал, що стосується роботи тестами.

Так, до основних функціональних вимог до розроблюваного програмного продукту можна віднести:

1) авторизація користувача у системі за допомогою свого облікового запису;
2) редагування користувачем своїх особистих даних, включаючи дані облікового запису;
3) створення викладачами тестів з наступними особливостями:
    - вибір типу: тренувальний чи контрольний;
    - можливість зазначення одного або декількох варіантів правильних відповідей;
    - можливість уточнення загального балу за питання (1 бал за замовчуванням);
    - опція «Виберіть усі правильні відповіді» замість конкретного числа правильних відповідей за замовчуванням;
    - обмеження дати проходження та для контрольних тестів часу проходження (тривалості);
    - зазначення певної структурної навчальної одиниці, для яких

тест буде доступний;

4) редагування існуючого тесту;
5) видалення існуючого тесту;
6) проходження тестів з автоматичною оцінкою результатів, що передбачає певні особливості відповідно до типу тесту та його налаштувань, серед найважливіших такі:
7) перевірка власних відповідей до питання для тренувальних тестів, тобто користувач може подивитися, які з обраних ним варіантів правильні, а які — ні;
8) для контрольних, де встановлене обмеження по часу, у разі вичерпання часу система перериває проходження та підраховує бали за ті відповіді, які користувач встиг дати.
9) можливість проходження тренувальних тестів в умовах відсутності підключення до БД (вивантаження та завантаження тестів у файлах в форматі XML в зашифрованому вигляді [10]);
10) перегляд успішності складання тестів;
11) перегляд статистики по тестам.

## 1.2 НЕФУНКЦІОНАЛЬНІ ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

На відміну від функціональних вимог, які визначають що система повинна робити, нефункціональні вимоги визначають якою система повинна бути. Узагальнено їх можна класифікувати на три категорії, вимоги до кожної з яких будуть детально перелічені нижче [1].

Вимоги до інтерфейсу:

— для взаємодії системи з базою даних клієнта, в конфігураційному файлі має бути прописана адреса сервера для підключення.

Апаратні та програмні вимоги:

— ОС Windows;
— мінімальна версія платформи .Net Framework – 4;

— рекомендований мінімум: процесор з частотою 1 ГГц;
— мінімальний обсяг ОЗП – 1 ГБ;
— MySQL та її користувач з правами адміністратора.

Операційні вимоги:

— безпека та конфіденційність: розроблене програмне рішення має обмежувати доступ користувачів до будь-якої інформації по сторонніх тестів, а також забезпечувати шифрування тестів, що вивантажуються на цифровий носій для офлайн роботи програми;
— надійність: програмний продукт повинен вірно і зрозуміло відповідати на будь-які запити від користувача, має бути захищений від помилок і гарантувати стабільність роботи;
— правила перевірки: програма не повинна допускати введення даних, які можуть спричинити некоректну або непередбачувану роботу або порушити будь-яку із вище зазначених вимог;
— збереження даних: система повинна чітко інформувати користувача про можливе змінення даних (містити кнопки для застосування змін чи запитувати підтвердження у разі відсутності такої кнопки). Для всіх пов’язаних даних має гарантуватися цілісне збереження.

## 1.3 ОГЛЯД ТА АНАЛІЗ АНАЛОГІЧНИХ РІШЕНЬ

У сучасній індустрії інформаційних технологій пропонується багато рішень для надання можливості проведення тестування. Варто зауважити, що більшість з них розроблені для веб-середовища, а серед решти — переважна частина якісної продукції є платною.

Найголовнішою перевагою системи, що розробляється у даному дипломному проекті, є її потенційна незалежність від інтернету. Тобто один з варіантів використання програми — в межах локальної мережі, без необхідності зовнішнього підключення. Також користувачу надається можливість вивантажити собі на персональний комп’ютер (ПК) тренувальні

тести, щоб він зміг готуватися навіть знаходячись поза межею мережі, де розташована база даних, або у разі відсутності до неї підключення.

Далі будуть детально розглянуті найбільш вдалі та розповсюджені програмні продукти, які мають схоже призначення. Серед них: Візуальна студія тестування, Система тестування INDIGO, Тесторіум та Moodle.

Перша система, Візуальна студія тестування від MMIS Lab, має наступні модулі: редактор тестів, оболонку для тестування, результати тестування, адміністрування. Серед переваг даної системи можна виокремити створення складного тексту та наявність алгоритму, що враховує статистичну похибку вгадування правильних відповідей.

Система тестування INDIGO реалізує усі модулі, розглянуті вище, а також надає адміністратору додаткову можливість побудови звітів та аналізу статистики.

Тесторіум відрізняється від попередніх передусім тим, що вона є безкоштовною та розташована у веб і має централізовану єдину базу даних. Вона є досить популярним програмним засобом і підходить для тренувань, але зазвичай не використовується для проведення реального контролю навчальними закладами, що пов’язано з її архітектурою. Перевагою та основною причиною популярності цієї системи є простота її інтерфейсу та зрозумілість використання, а також доступність з будь-якого приладу з доступом до Інтернету, проте ця простота розповсюджується і на її функціональність, що не дає можливості створити складно сконфігуровані тести чи налаштування до них.

І нарешті система під назвою Moodle – це навчальне середовище, створене для управління курсами, яке надає викладачам, учням та адміністраторам дуже розвинутий набір інструментів для комп'ютеризованого навчання. Серед компонентів Moodle є і модуль тестування. Лише ця система серед розглянутих рішень є локальною та безкоштовною, але вона є комплексним засобом, що включає в себе багато непов’язаних з тестуванням модулів, тому для такого вузького призначення є незручною.

Підсумок узагальненого порівняння розглянутих програмних продуктів представлений у таблиці 1.1.

Таблиця 1.1 — Порівняльна характеристика існуючих рішень

| Критерій | Візуальна студія тестування | Система тестування INDIGO | Тесторіум | Moodle |
|---|---|---|---|---|
| Український інтерфейс | - | + | + | + |
| Робота офлайн | - | - | - | - |
| Можливість лише локального підключення | + | - | - | + |
| Автономне рішення | + | + | + | - |
| Різноманітність параметрів тестів | + | + | - | + |
| Вільна ліцензія | - | - | + | + |

Бачимо, що серед популярніших рішень лише два підтримують можливість локального розміщення програми: Візуальна студія тестування та Moodle. Проте перша не є безкоштовною, а Moodle є комплексним програмним забезпеченням, тестування в якому є одним з модулів організації курсів, тому при необхідності задоволення лише однієї цілі — проведення тестування — його використання потребує зайві апаратні ресурси та надлишкові дії користувача.

## 1.4 ВИСНОВКИ

В даному розділі пояснювальної записки були детально описані функціональні та нефункціональні вимоги до системи тестування, а також проаналізовані існуючі аналогічні рішення.

Так, оцінивши, згідно з таблицею 1.1, переваги та недоліки кожного з розглянутих рішень та врахувавши фактори, що грають найважливішу для потенційних користувачів, було зроблено висновок, що застосунок, який розробляється, має задовольняти наступним критеріям:

— робота програми з локальним сервером, на якому розташована БД;
— зручний інтерфейс;
— простота й зрозумілість використання;
— розмаїття налаштувань, параметрів та обмежень для тестів;
— відображення у грі поточного стану, результатів і досягнень користувача
— можливість підготовки вихованцями до контрольних тестів без доступу до сервера БД.

Отже, згідно з результатами, отриманими у цьому розділі, були остаточно визначені вимоги до розроблюваного програмного забезпечення для навчальних закладів.

# 2 ВИБІР ТА ОБГРУНТУВАННЯ КОМПОНЕНТІВ СИСТЕМИ

## 2.1 ЦІЛЬОВА ПЛАТФОРМА

Метою дипломного проекту є розробка настільної програми для операційної системи Microsoft Windows. Вибір цієї ОС обумовлений передусім тим, вона займає близько 79% ринку для застосунків такого типу у світі [2], а в Україні її частка сягає 86%.

Серед переваг Windows можна виділити наступні:

— під керуванням цієї ОС працюють неймовірна кількість пристроїв, практично будь-яка програма або її аналоги;
— апаратна та програмна сумісність;
— повне використання апаратних ресурсів;
— багатозадачність;
— захищений режим;
— підтримка та легкість встановлення;
— обмін даними між застосунками;
— функціональність та зручний та інтуїтивно зрозумілий інтерфейс;
— легке відновлення видаленої інформації.

Програмне забезпечення, виконане в рамках даного дипломного проекту, розроблене за допомогою програмної платформи .NET Framework від компанії Microsoft — технології, що розрахована на роботу під ОС Microsoft Windows. Ця платформа представляє собою всебічну і узгоджену модель програмування для побудови застосунків, що володіють привабливим інтерфейсом користувача, прозорими і безпечними засобами зв'язку, а також можливістю створення різноманітних бізнес-процесів.

Дана платформа представлена у декількох версіях, для розроблюваної системи мінімально необхідною версією була обрана 4.0, випущена у 2010 році, що підходить навіть для Windows XP.

Мінімальні рекомендовані апаратні вимоги для .NET Framework 4.0:

1) процесор Pentium з тактовою частотою 1 ГГц;
2) 512 МБ оперативної пам’яті;
3) мінімальне місце на диску:
   — x86 – 850 МБ;
   — x64 – 2 ГБ.

Microsoft .NET Framework 4 працює разом зі своїми попередніми версіями. Застосунки, засновані на попередніх версіях .NET Framework, будуть продовжувати виконуватися на платформі, для якої вони призначені за замовчуванням.

Платформа містить наступні нові можливості і удосконалення [3], які використовувалися під час розробки програми:

— вдосконалення в CLR (Common Language Runtime) і BCL (Base Class Library);
— нововведення в мові C#, наприклад лямбда-оператори, неявні продовження рядків, а також іменовані і необов'язкові параметри.
— удосконалення в доступі до даних і моделюванні;
— вдосконалення в Windows Presentation Foundation (WPF).

## 2.2 МОВА ТА ТЕХНОЛОГІЯ ПРОГРАМУВАННЯ

Для розробки програмного забезпечення під платформу .NET була обрана така високорівнева мова програмування, як C# [4]. Мова C# розроблена під егідою Microsoft, об’єктно-орієнтована мова з безпечною системою типізації.

Прийнято виділяти наступні найважливіші переваги, які надає дана мова:

— автоматичне керування пам'яттю;
— можливість створення багатопотокових застосунків;
— підтримка узагальнень;
— підтримка лямбда виразів та замикань, а також функціонального програмування;
— розширені можливості обробки виняткових ситуацій;

— механізм абстракцій, наслідування, поліморфізм, інкапсуляція;

— вказівники на функції-члени класів, делегати;

— велика стандартна бібліотека класів, зокрема для роботи з файлами та колекціями;

— коментарі у форматі XML.

В якості технології для розробки системи була обрана така система побудови клієнтських застосунків, як Windows Presentation Foundation [5, 6]. В основі цієї технології лежить векторна система візуалізації, що не залежить від розрішення пристрою виведення. Графічної технологією, що лежить в основі WPF, є DirectX [7], що забезпечує порівняно вищу продуктивність WPF за рахунок використання апаратного прискорення графіки.

WPF забезпечує розробника наступними засобами для створення візуального інтерфейсу:

— різноманітні елементи управління;

— використання сторінок для проектування браузеро подібних застосунків;

— прив'язка даних моделі до представлення;

— композитність (вкладеність) елементів управління;

— макети, шаблони та стилі;

— двомірна і тривимірна графіка;

— анімація, аудіо та відео;

— ресурси, що дозволяють об’єднати різні інтерактивні об’єкти в межах компоненту (вікна, панелі чи окремого елементу) для спрощення роботи з ними;

— документи, текст, мультимедіа і оформлення;

— XPS документація.

Ця технологія дозволяє використовувати мову C# для створення логіки застосунку, а для графічного інтерфейсу — eXtensible Application Markup Language (XAML). XAML — це мова декларативного опису інтерфейсу, що

базується на XML. Вона реалізує модель поділу коду та дизайну, що дозволяє кооперуватися програмісту і дизайнеру.

## 2.3 СИСТЕМА КЕРУВАННЯ БАЗАМИ ДАНИХ

Розроблена програма використовує таку реляційну [8] систему управління базами даних, як MySQL. Ця СКБД розробляється і підтримується корпорацією Oracle та має вільну ліцензію. За статистикою вона є найпопулярнішою базою даних з відкритим кодом у світі.

MySQL базується на мові SQL і призначена для малих та середніх програмних рішень [9, 10]. SQL як частину системи MySQL можна охарактеризувати як мову структурованих запитів плюс найбільш поширена стандартна мова, яка використовується для доступу до баз даних. Зазвичай вона використовується як сервер, до якого звертаються локальні або віддалені клієнти, проте в дистрибутив входить бібліотека внутрішнього сервера, що дозволяє включати MySQL в автономні програми.

MySQL має API та конектори [11] для більшості мов програмування, бібліотеки для мов платформи .NET, а також забезпечує підтримку для ODBC за допомогою ODBC-драйвера MyODBC.

Сильними сторонами даної СКБД вважаються:

— простота встановлення та використання;
— малий розмір;
— необмежена кількість користувачів, що одночасно працюють із БД;
— найкраща швидкість обробки даних на обсязі до 500000 записів;
— високий рівень безпеки доступу до даних та шифрування при передачі;
— легка розширюваність;
— апаратна сумісність.

До основних можливостей MySQL відносять кеш запитів, повнотекстовий пошук, управління транзакціями та блокування, функції, тригери,

представлення, індекси і різноманітні типи даних. Також варто зазначити, що MySQL містить велику кількість типів таблиць: користувачі можуть обрати як таблиці типу MyISAM, що підтримують повнотекстовий пошук, так і таблиці InnoDB, що підтримують транзакції на рівні окремих записів.

MySQL підтримує наступний набір вбудованих типів даних:

— чисельні типи (цілі різного розміру, з фіксованою точкою, з плаваючою точкою);
— символьні типи фіксованої чи довільної довжини;
— двійкові типи (включаючи BLOB);
— типи «дата/час», що повністю підтримують різноманітні формати, точність, формати виводу, включаючи останні зміни у часових поясах;
— перерахування;
— множини.

## 2.4 ВИСНОВКИ

В даному розділі було надано опис та обґрунтування компонентів системи та технології, що застосовуються під час її розробки.

Так, в якості мінімальної платформи для ОС Microsoft Windows була обрана .NET Framework 4. Вона включає в себе сучасну графічну технологію для побудови клієнтських застосунків WPF, за допомогою якої і було розроблене дане програмне забезпечення. В межах даної технології бізнес-логіка реалізується за допомогою високорівневої об’єктно-орієнтованої мови програмування C#, а мова розмітки XAML застосовується для декларативного опису графічного інтерфейсу програми.

Також було зазначено, що застосунок передбачає взаємодію з такою СКБД, як MySQL, оскільки вона має вільну ліцензію та володіє усіма необхідними засобами та характеристиками, щоб задовольнити потреби системи.

# 3 АРХІТЕКТУРА СИСТЕМИ

## 3.1 ДВОРІВНЕВА АРХІТЕКТУРА «КЛІЄНТ — СЕРВЕР»

Система тестування для навчальних закладів представлена дворівневою архітектурою та складається з клієнтського застосунку та сервера, на якому розташована база даних, з якого взаємодіє застосунок. Схема такої архітектури представлена на рисунку 3.1.

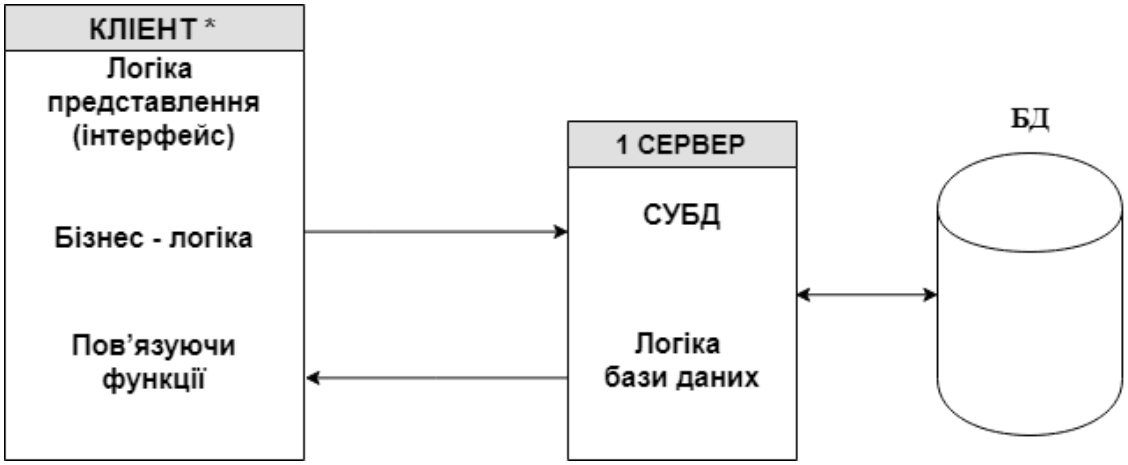



Рисунок 3.1 — Дворівнева архітектура «Клієнт — Сервер»

У такій моделі база даних зберігається на сервері. Там же знаходиться ядро СКБД. На клієнті розташовується презентаційна логіка і бізнес-логіка програми. Клієнт звертається до сервера з запитами на мові SQL [12].

Фактично клієнт і сервер - це електронні обчислювальні машини, на першій з яких розташований клієнтський застосунок, в той час як на другій — система керування базою даних. Клієнт та сервер зазвичай розташовані на різних машинах, але можуть бути і на одній машині.

Ці два компоненти системи взаємодіють між собою через обчислювальну мережу, підключення до серверної БД від клієнта здійснюється за допомогою спеціального конектора.

Сервер може приймати запити від множини клієнтів. У зв'язку з цим, СКБД зазвичай розміщують на спеціально виділеній обчислювальній машині, продуктивність якої повинна бути високою.

Перевагами такої архітектури є:

— вимоги до машин, на яких працюватиме клієнт, знижуються, оскільки частина обчислень виконуються на сервері;

— усі дані зберігаються на сервері, який, як правило, захищений набагато краще за більшість клієнтів;

— захист даних засобами СКБД, що дозволяє блокувати недозволені користувачеві дії;

— сервер реалізує управління транзакціями і може блокувати спроби одночасного зміни одних і тих же записів;

— на сервері простіше організувати контроль повноважень, щоб вирішувати доступ до даних тільки клієнтам з відповідними правами доступу.

Також для зменшення навантаження на клієнта, в даному проекті передбачено винесення частини бізнес-логіки на сервер. Для цього в логіку СКБД додаються тригери для перевірки коректності даних та збережені процедури — спеціальних програмні модулі — для повернення клієнту релевантних його запиту даних, які потребуються певних обчислень або обробки та потрібні клієнту або для виведення на екран, або для виконання частини бізнес-логіки, яка розташована на клієнті.

Таким чином, замість потужностей клієнтських машин використовуються потужності сервера. Така модель називається моделлю «активного серверу БД».

Як і будь-яка інша архітектура, «клієнт-сервер» не позбавлена недоліків. Так, недоліками можна вважати наступні;

— непрацездатність сервера може зробити непрацездатною всю систему;

— підтримка роботи системи вимагає окремого фахівця - системного адміністратора;

— розосередження логіки системи між її компонентами у випадку

активного серверу БД, що ускладнює внесення змін до системи.

## 3.2 ШАБЛОН ПРОЕКТУВАННЯ MODEL-VIEW-VIEWMODEL

Як було описано у розділі 2 даної пояснювальної записки для розробки програми було вирішено застосувати сучасну платформу розробки від Microsoft WPF. Саме на цю платформу орієнтований такий шаблон проектування, як MVVM [13].

Цей шаблон полегшує відокремлення розробки візуальної частини, тобто графічного інтерфейсу, від розробки бізнес-логіки, також відомої як модель (можна сказати, що це відокремлення представлення від моделі). Необхідністю цього є надання можливості змінювати їх незалежно одну від одної.

В шаблонах-попередниках, наприклад Model-View-Controller та Model-View-Presentation, зміни у представленні користувача не мали жодного безпосереднього впливу на модель, а відбувалися за допомогою компонента Controller чи Presenter. У WPF, в свою чергу, присутня нова концепція «зв'язування даних», що дозволяє прив'язувати дані моделі до візуальних елементами в обидві сторони.

MVVM складається з трьох компонентів, представлених на рисунку 3.2: моделі (Model), моделі подання (ViewModel) і представлення (View).

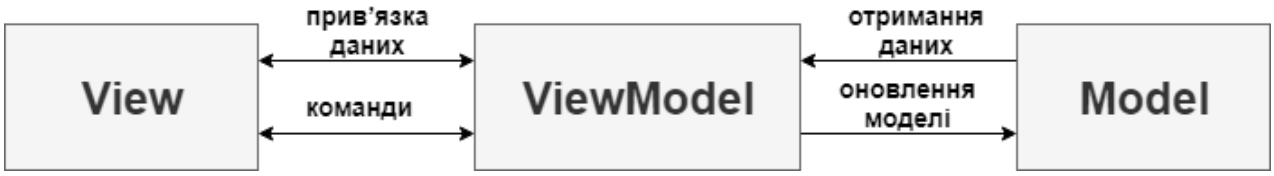



Рисунок 3.2 — Архітектурний шаблон Model-View-ViewModel

Модель являє собою фундаментальні дані, що використовуються під час роботи застосунку. Моделі можуть містити логіку, безпосередньо пов'язану цими даними, наприклад, логіку валідації властивостей моделі чи функції, які відповідають її поведінці. У той же час модель не повинна містити ніякої логіки,

пов'язаної з відображенням даних і взаємодією з візуальними елементами управління.

Згідно з загальновизнаною практикою моделі в даному дипломному проекті реалізують інтерфейси INotifyPropertyChanged та для колекцій — INotifyCollectionChanged, які дозволяють повідомляти систему про зміни властивостей моделі.

Представлення — це графічний інтерфейс, через який користувач взаємодіє з застосунком. У WPF — це код в XAML, який визначає інтерфейс у вигляді кнопок, текстових полів та інших візуальних елементів.

В ідеалі код представлення C # не повинен містити якийсь логіки, крім конструктора, який викликає метод InitializeComponent і виконує початкову ініціалізацію вікна, оскільки вся основна логіка виноситься в компонент ViewModel. Однак іноді в файлі пов'язаного коду все може знаходитися певна логіка, яку важко реалізувати в рамках шаблону MVVM у ViewModel.

Модель представлення (ViewModel, що означає «Model of View») пов'язує модель і уявлення через механізм прив'язки даних.

З одного боку є абстракцією Представлення, а з іншого надає обгортку даних з Моделі, які мають зв'язуватись. Тобто вона містить Модель, яка перетворена до Представлення, а також містить у собі команди, якими може скористатися Представлення для впливу на Модель. З цієї точки зору, модель представлення більше схожа на модель, ніж на представлення і обробляє більшість, якщо не всю, логіку відображення даних.

Якщо в моделі змінюються значення властивостей, при реалізації моделлю інтерфейсу INotifyPropertyChanged автоматично йде оновлення відображуваних даних в поданні і навпаки: зміна даних у графічних елементах призводить до оновлення значень властивостей моделі, хоча безпосередньо модель і уявлення не пов'язані. Фактично ViewModel призначена для того, щоб:

— здійснювати зв'язок між моделлю та вікном;

— відслідковувати зміни в даних, зроблені користувачем;

— відпрацьовувати логіку роботи Представлення (механізм команд).

## 3.3 ОПИС МОДУЛІВ СИСТЕМИ

До складу системи тестування, що розроблюється у ході дипломного проекту, можна віднести наступні компоненти:

— модуль входу до системи;
— стартове меню;
— редактор особистих даних користувача;
— редактор тестів, що використовується для створення тестів та внесення змін;
— модуль проходження тестів;
— модуль експорту та імпорту тренувальних тестів на носій;
— модуль статистики.

Редактори даної системи мають наступну особливість реалізації: зберігання в пам’яті вхідного об’єкта та його копії, що обумовлено застосуванням шаблону MVVM. Оскільки даний архітектурний шаблон передбачає прив’язку даних моделі до інтерфейсу, то зміни графічних компонентів системи безпосередньо призводять до зміни моделі, тому усі оновлення зберігаються у копії до моменту явного виклику користувачем операції «Зберегти». Після ініціалізації такої дії копія синхронізуються з вхідним об’єктом. Якщо вікно не містить відповідної кнопки для збереження, це означає, що зміни автоматично будуть застосовані до вхідного об’єкту.

Далі в рамках цього розділу буде детально розглянуто функціонал та принцип реалізації кожного з перелічених модулів.

### 3.3.1 Модуль входу до системи та стартове меню

Даний модуль використовується одразу після запуску системи і призначений для входу зареєстрованих користувачів до системи. До даного модулю відноситься вікно авторизації, що має фіксований розмір і зображене на рисунку 3.3.

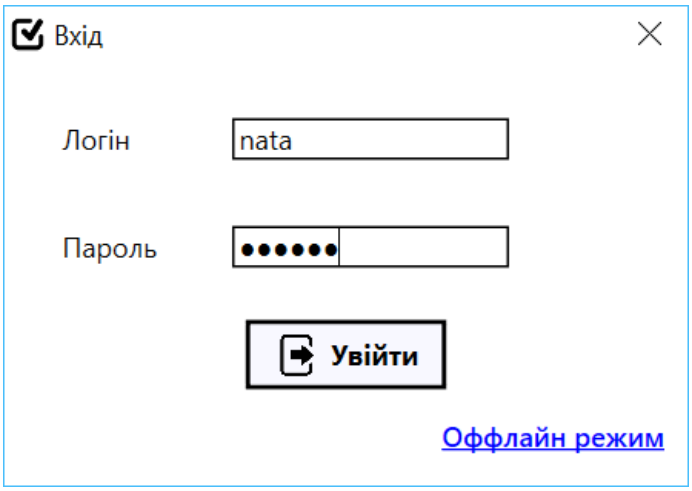



Рисунок 3.3 — Вікно авторизації

Вхід до системи здійснюється по кнопці «Увійти» за допомогою облікового запису, що містить інформацію про логін та пароль користувача. Логін користувача має бути унікальним в рамках локального екземпляру БД, а паролі зберігаються у БД у виглядів гешів, отриманих за допомогою такого алгоритму, як MD5 [14].

Вікно входу здійснює перевірку текстових полів на наявність введених даних та підсвічує червоним кольором незаповнені поля. У разі введення невірного логіна чи пароля система повідомляє користувача про конкретну помилку за допомогою діалогового вікна з повідомленням.

Власне сам обліковий запис користувача призначений для ідентифікації особи у системі та збереження пов’язаних з ним даним. Таким чином, підтримується конфіденційність та цілісність даних: користувач може побачити лише свої об’єкти, а самі об’єкти завжди можна ідентифікувати по наявним зв’язкам.

Альтернативний варіант входу до системи — по мітці «Оффлайн режим», який призначений для роботи з системою без підключення до БД.

Практичне значення такого режиму полягає в уможливленні підготовки вихованців до контрольних тестів шляхом проходження тренувальних тестів, які були попередньо вивантажені з БД на цифровий носій інформації. В такому режимі дії користувача не зберігається, тобто результати проходження

тренувальних тестів не будуть занесені до БД і відповідно відобразяться в подальшому у модулі статистики.

У разі успішного входу користувач побачить стартове меню (рисунок 3.4), що відображає основні доступні функції, представлені у вигляді кнопок-зображень з підказками.

Повний перелік пунктів стартового меню:

1) обліковий запис;
2) створення тесту;
3) редагування тесту;
4) проходження тесту;
5) статистика;
6) вихід.


Рисунок 3.4 — Стартове меню

Насиченість цього меню залежить перш за все від типу користувача: викладач чи вихованець. Так, для викладача передбачені усі функції, крім

проходження тесту, а для вихованця відсутні модулі створення та редагування тестів. Для офлайн режиму дане меню містить лише один пункт – проходження тесту.

### 3.3.2 Модуль редагування особистих даних користувача

Модуль редагування особистих даних користувача — редактор особистих даних — графічно представляє собою вікно з вкладками для редагування особистих даних особи, яке доступне з пункту меню 1 (дивись 3.3.1). Вікно цього редактору зображене на рисунку 3.5.

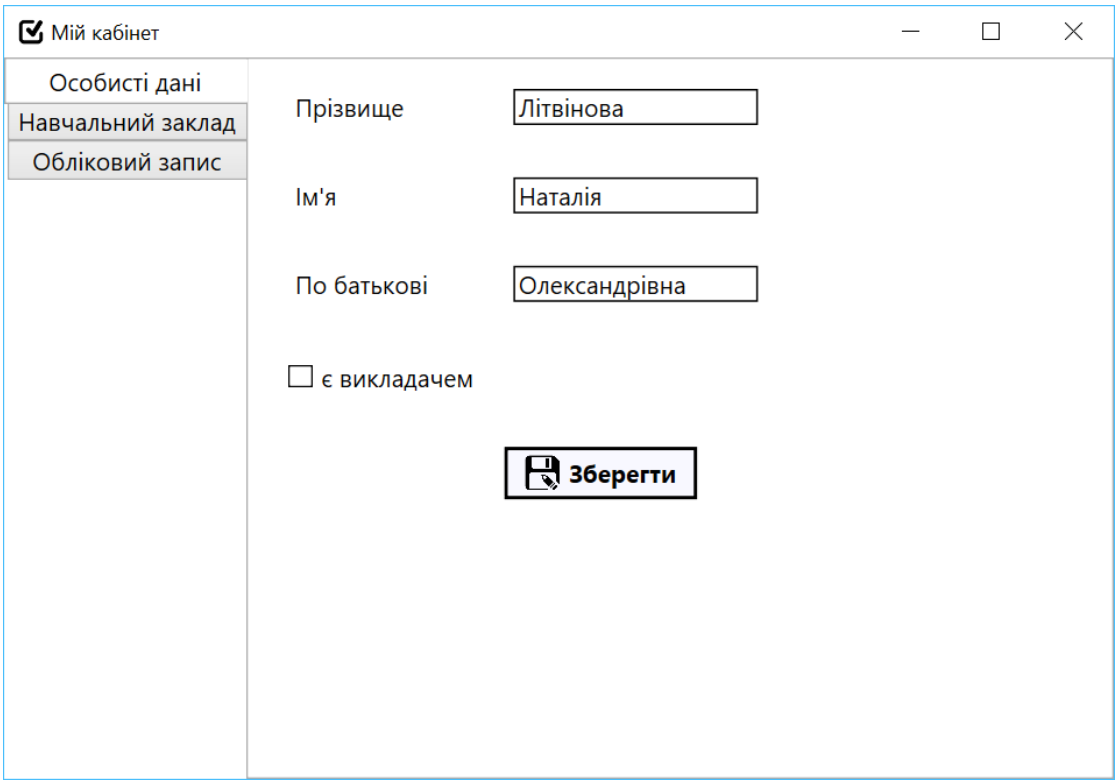



Рисунок 3.5 — Вікно редактору особистих даних

Редактор надає користувачу системи можливість внесення змін до представлених там даних і розподіляє їх на три основні блоки:

— редактор особистих даних, який містить графічні елементи для таких даних, як: прізвище, ім'я, по батькові особи, а також ознаку чи є дана особа викладачем;

— редактор структурних навчальних одиниць, до яких належить дана особа, представлений у вигляді таблиці;

— редактор даних облікового запису що відображає логін користувача та надає можливість змінити пароль до облікового запису.

Збереження даних на першій вкладці відбувається після натискання ЛКМ по кнопці «Зберегти», а на решті вкладок – автоматично після підтвердження дії користувачем.

### 3.3.3 Модуль управління тестами

Модуль управління тестами — редактор тестів —надає єдиний інтерфейс для створення і редагування тестів. Для організації та об’єднання пов’язаних даних використовується TabControl, що за замовчування містить лише вкладку для загальної інформації (рисунок 3.6).

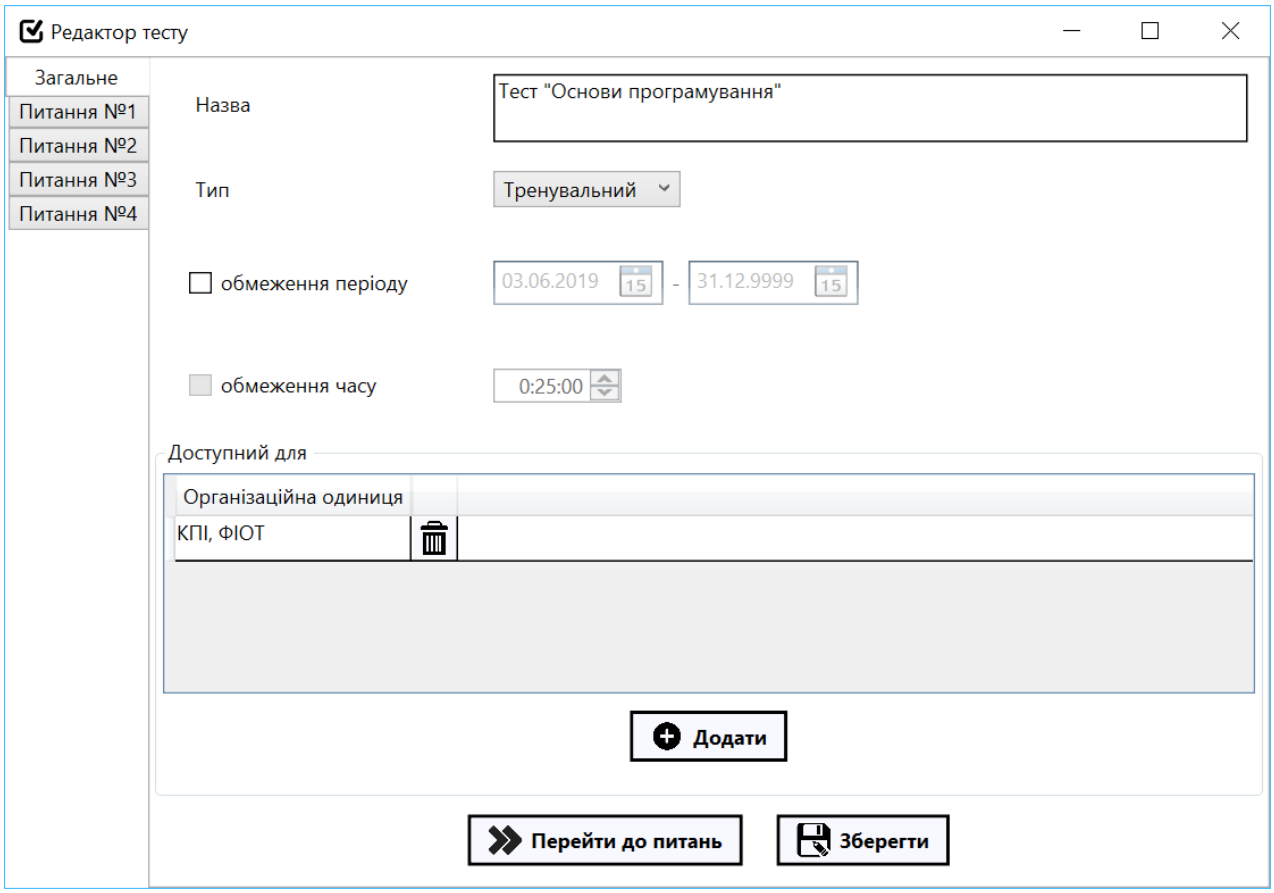



Рисунок 3.6 — Редактор тесту. Загальна інформація

На вкладці «Загальне» викладач має можливість:

— зазначити назву тесту;

— обрати тип тесту: тренувальний чи контрольний;

— обмежити тест за датою проходження, вибравши з календаря дві дати, що задають період, коли тест можливо буде пройти;

— обмежити час проходження тесту вихованцями лише для контрольних тестів;

— назначити доступність тесту, зазначивши навчальні відділи, які матимуть доступ до цього тесту.

Кнопка «Перейти до питань» слугує для переходу на вкладки, що містять інформацію по питанням. Такі вкладки формуються та додаються динамічно: для існуючого тесту одразу при відкритті редактора, а для нового після натискання кнопки «Перейти до питань» по спеціальному шаблону. Приклад такої вкладки можна побачити на рисунку 3.7.

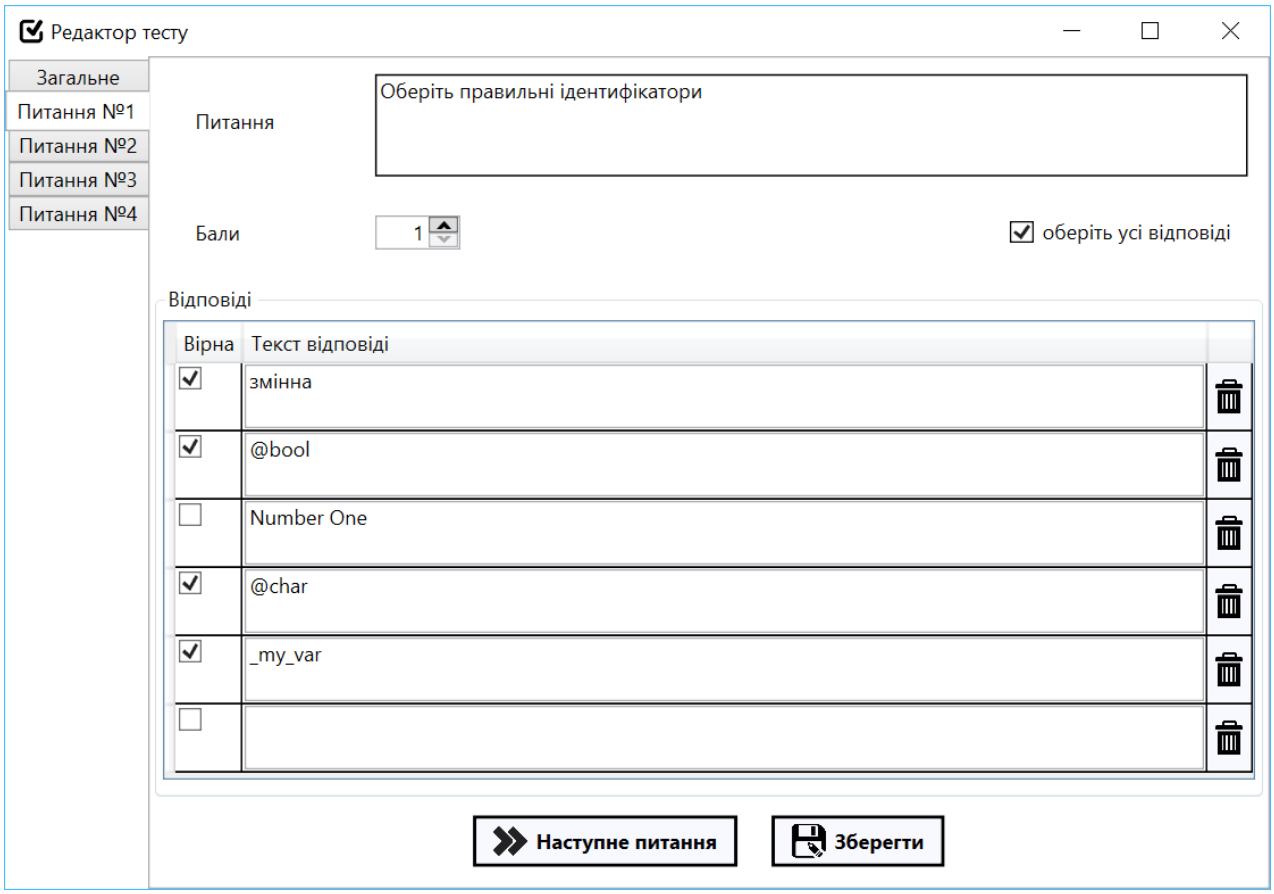



Рисунок 3.7 — Редактор тесту. Питання та відповіді

Як видно на рисунку 3.6, створена по шаблону для питань вкладка надає користувачу наступні можливості:

— зазначити заголовок питання;
— вказати відмінну від одиниці сумарну кількість балів за дане питання;
— активувати опцію «Оберіть усі відповіді», що скасовує відображення вихованцю кількості правильних відповідей на питання та дещо змінює алгоритм підрахунку балів за питання (дивись розділ 5.2);
— редагувати варіанти відповідей: їх кількість, текст відповіді та її правильність;
— перейти до наступного питання (якщо такого нема, буде створена нова вкладка).

### 3.3.4 Модуль експорту та імпорту тестів на носій

Однією з функціональних вимог до розроблюваної системи є можливість підготовки вихованцями до тестів в умовах відсутності доступу до сервера БД, так званий режим офлайн. Як вже зазначалося, це уможливлюється наявністю локальної копії тренувальних тестів.

Для забезпечення функціонування офлайн режиму був розроблений модуль експорту та імпорту тестів на носій, який доступний з пункту меню 4 (дивись 3.3.1). На рисунку 3.8 показана кнопка за допомогою якої можна вивантажити тренувальний тест на електронний носій інформації, а на рисунку 3.9 — завантажити назад у систему.

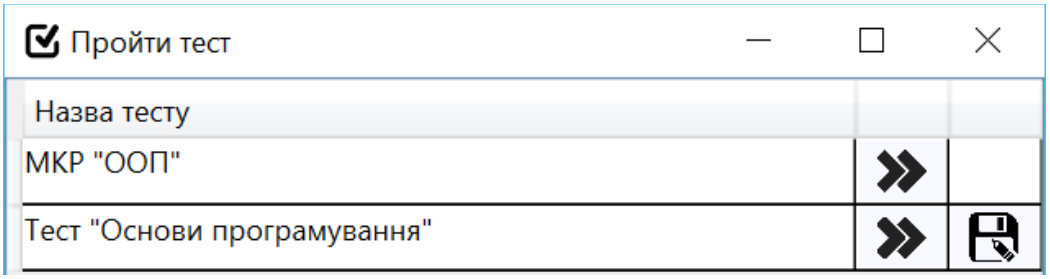



Рисунок 3.8 — Кнопка збереження тесту на носій (справа)

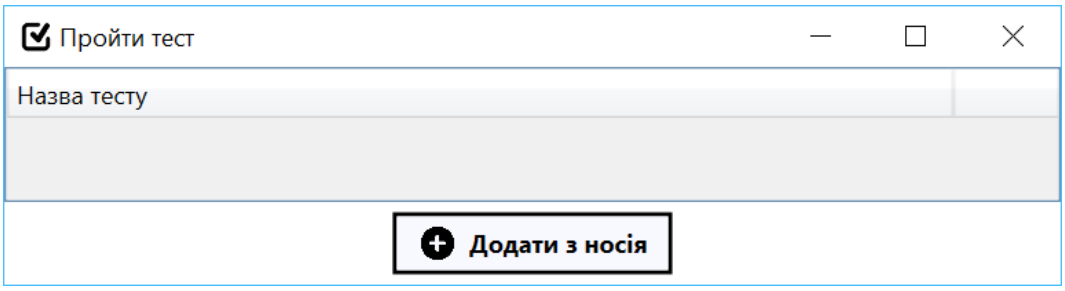



Рисунок 3.9 — Кнопка для додання файлу тесту з носія

Так, користувач має завчасно імпортувати (вивантажити) тест собі на електронний носій, коли програма має доступ до серверу БД та зв'язок з самою БД для того, що отримати інформацію по тесту. Дані щодо тесту зберігаються у текстовому файлі в форматі XML, а потім шифруються за допомогою симетричного алгоритму AES [15, 16] для уникнення компрометації. Потім, маючи такий файл, користувач може увійти у систему в офлайн режимі, що було описано у розділі 3.3.1, та імпортувати файл у систему. Модуль експорту та імпорту розшифрує файл та на основі даних, що зберігаються в XML, зчитує дані про тест та завантажує сформований на її основі об'єкт тесту у пам'ять. Завантажені тести будуть зберігатися у програмі до моменту виходу з нею, що надасть можливість користувачу повторно проходити тести для тренувань.

Варто зауважити, що експортувати можна лише тренувальні тести, для яких в XML буде записана наступна інформація:

— назва тесту;
— обмеження періоду проходження тесту;
— питання з ознакою «Оберіть усі відповіді» за її наявності;
— варіанти відповідей з зазначення правильних.

Приклад XML пакету можна переглянути у додатку А.

### 3.3.5 Модуль проходження тестів

Для проходження вихованцями тестів був розроблений окремий модуль, який надає єдиний інтерфейс для проходження як тренувальних, так і

контрольних тестів незалежно від встановлених для них обмежень. Приклад вікна проходження тесту зображений на рисунку 3.10.

Серед особливостей інтерфейсу вікна:

— відображення вкладок з вказанням номеру питання;

— наявність варіантів відповідей у вигляді списку з графічними елементами «прапорцями» для вибору відповідей зліва;

— зазначення кількості правильних відповідей поряд з міткою «Оберіть» або напису «усі», якщо це питання з опцією «Оберіть усі правильні відповіді»;

— зазначення максимальної кількості балів за питання справа від заголовку питання;

— функціональні кнопки під питанням для навігації та можливої перевірки даних відповідей по цьому питанню;

— вказання кількісті завершених питань.

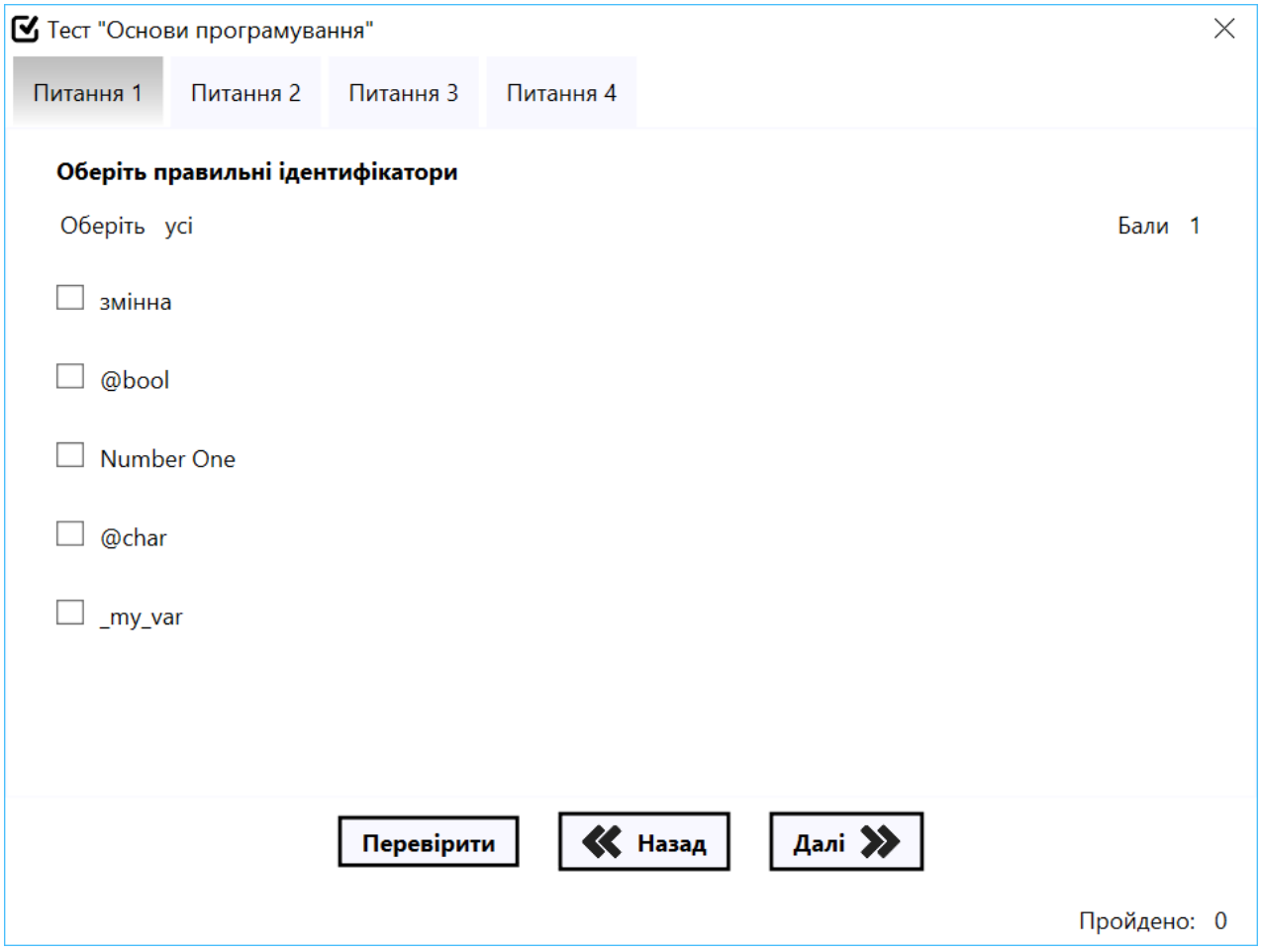



Рисунок 3.10 — Вікно проходження тесту

Для організації проходження тесту та оцінки результатів в даному модулі реалізовано такі функціональні можливості:

— контроль кількості обраних відповідей на питання: було вирішено допускати меншу за необхідну кількість, проте для звичайних питань, де задана кількість правильних відповідей, система не допускає вибір більшого за правильний числа варіантів;
— навігація між питаннями за допомогою кнопок «Назад» та «Далі»;
— завершення тесту за допомогою кнопки «Далі», якщо користувач «знаходиться» на останньому питанні тесту;
— перевірка при завершенні тесту чи на всі питання була дана хоч одна відповідь: якщо ні, то модуль повідомляє користувача про таке питання за допомогою діалогового вікна та активую вкладку з цим питанням;
— можливість перевірити обрані користувачем відповіді до активного питання для тесту типу «Тренувальний» шляхом натискання ЛКМ по кнопці «Перевірити».

### 3.3.6 Модуль статистики

Модуль статистики доступний для обох ролей користувачів, але його робота і відображувана інформація значно відрізняється в залежності від ролі. Тому далі буде розглянуто доступні функції окремо для викладачів та окремо для вихованців.

Викладач за допомогою цього модуля може:

— переглянути оцінки, які отримали його вихованці за створені ним контрольні тести;
— відкрити детальну інформацію по таким тестам для перегляду відповідей конкретного вихованця;
— з’ясувати загальну кількість проходжень по його тестам, а також кількість проходжень унікальними користувачами.

Для вихованця модуль статистики надає наступні можливості:

— переглянути оцінки за контрольні тести;

— побудувати графік успішності проходжень тренувальних тестів.

## 3.4 ВИСНОВКИ

Таким чином для системи тестування для навчальних закладів була обрана дворівнева архітектура, що передбачає наявність клієнту та серверу БД. Основними перевагами, які забезпечує дана архітектура є: зниження вимог до машин, на яких працюватиме програма-клієнт та зберігання даних на сервері, який є більш надійним та захищеним.

Підсумком застосування шаблону MVVM є відокремлення розробки графічного інтерфейсу від розробки бізнес-логіки, а також функціональний розподіл програми на компоненти, які простіше розробляти, а також в подальшому модифікувати і підтримувати, використовуючи механізм «прив’язки моделі до представлення.

Також в даному розділі були детально розглянуті усі модулі системи з зазначення їх призначення, принципу роботи, наведенням опису їх основних функціональних можливостей та прикладів інтерфейсу.

# 4 РЕАЛІЗАЦІЯ МОДЕЛІ ДАНИХ

## 4.1 АНАЛІЗ ІНФОРМАЦІЙНИХ ПРОЦЕСІВ

Інформаційні потреби системи тестування невід'ємно зв'язані з об'єктами предметної області та визначаються перш за все її користувачами: викладачами та вихованцями.

Джерелом інформації системи служить реляційна база даних, яка розташована на спеціально відведеній машині — сервері. Приймачами ж інформації є власне самі користувачі. Ними, як зазначалося, є співробітники закладів, що надають освітні послуги, та їх клієнти.

Основні інформаційні процеси слідують з інформаційних потреб користувачів і визначаються переважно функціональними та нефункціональними вимоги, описаними у розділі 1.

Щодо інформаційних процесів, визначених функціональними вимогами до системи, то передбачені наступні:

— передача даних між БД та застосунком;
— передача і збереження даних до стороннього файлу;
— додавання нових записів до таблиць БД;
— видалення чи змінення існуючих у таблицях записів;
— обробка та перевірка введених користувачем даних;
— формування вибірки з таблиці з застосуванням умов фільтрації та сортування;
— збирання та поєднання даних з декількох таблиць;
— зв'язування даних за допомогою зовнішніх ключів у таблицях;
— перетворення інформації: зміна типу, формату.

Нефункціональні вимоги, зокрема, операційні, визначають такі інформаційні процеси, як:

— вплив інформації, що знаходиться в межах клієнтського застосунку, на таблиці у БД: наприклад, коли вихованець завершує проходження

тесту, то його результати одразу додаються до таблиці, що містить загальні відомості по проходженню тесту та до таблиці, яка зберігає відповіді користувача в рамках цього проходження;

— підтримка достовірності інформації: зміни інформації про об’єкт, його властивості та зв’язки, що містяться у БД, автоматично оновлюються у застосунку;
— підтримка цілісності: інформація, яка пов’язана з іншими об’єктами, не може бути видалена, що досягається наявністю тригерів з умовами на операції оновлення та видалення;
— підтримка надійності: введена інформація має піддаватися перевірці, як зі сторони клієнта, так і засобами СКБД на сервері, для уникнення нелогічних зв’язків та некоректних даних, вона має зберігатися лише після підтвердження правильності.

## 4.2 ПРОЕКТУВАННЯ БАЗИ ДАНИХ

До складу розробленої бази даних входять 18 таблиць та збережені процедури, які використовуються клієнтами для отримання вибірки з таблиць, яка потребує складних обчислень.

Серед 18 таблиць 8 являються основними, тобто представляють моделі об’єктів предметної області, і будуть детально розгляну у 4.2.1, а решта є допоміжними для зберігання службової організації та організації механізму зв’язування записів та доповнення довільною кількістю необов’язкових атрибутів.

Усі таблиці з колонками та відношення між таблиця зображені на ER-діаграмі [17], що міститься на кресленику ІТ51.12.0БАК.005 Д3.

Для початку, буде розглянуто таблиці БД, які використовуються для організації різного роду зв’язків між записами в різних таблицях (дивись рисунок 4.1). Так, для цієї мети було створено системний реєстр (sys_reestrs),

що містить записи усіх основних таблиць БД, та класифікатор зв’язків (cl_links), який зберігає ідентифікатори та назви можливих видів зв’язків.

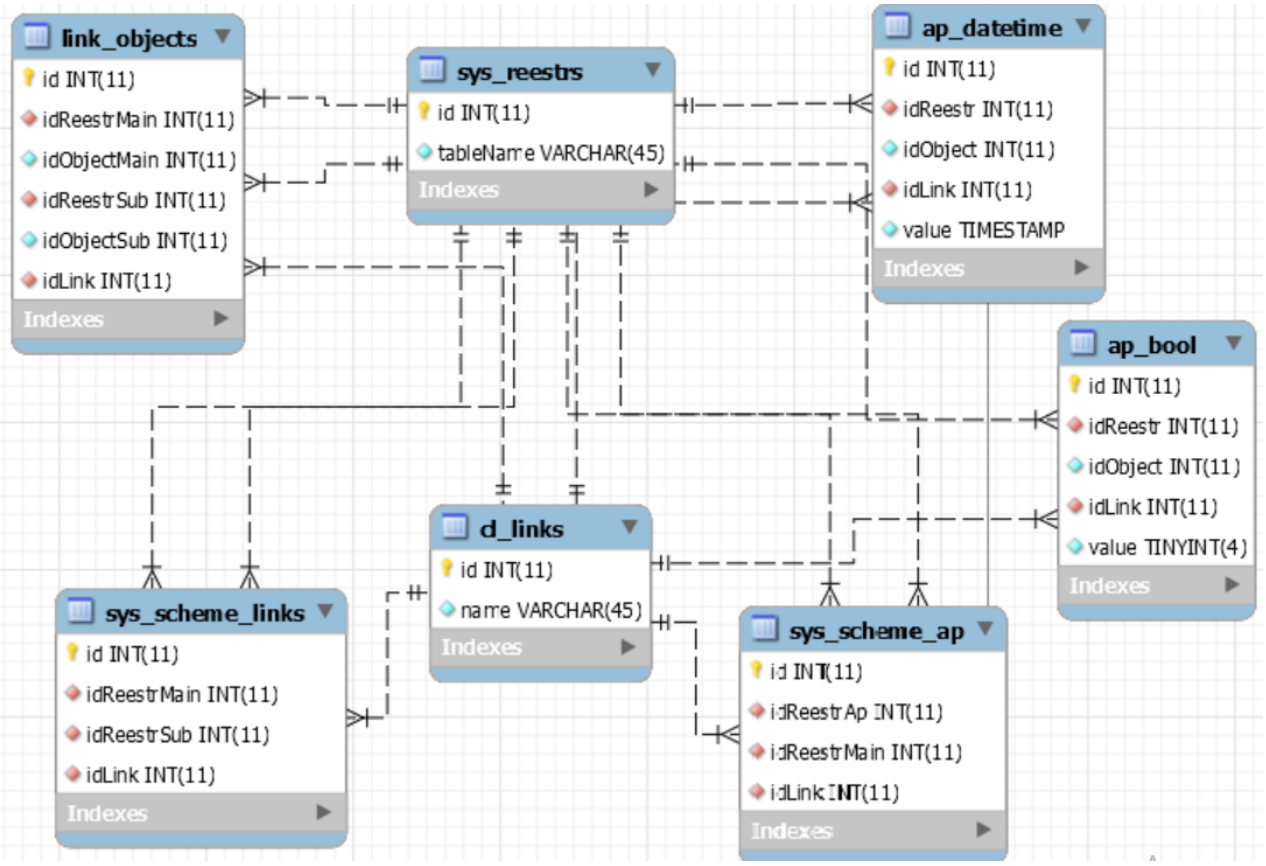



Рисунок 4.1 — Організація зв’язків між таблицями та додаткових даних

Таким чином, якщо необхідно зв’язати, наприклад, тест з питання , то до таблиці зв’язків об’єктів вноситься запис з даними тесту як головного об’єкту (його id та id реєстру тест, взятого з системного реєстру) та аналогічні дані для відповідного питання, та визначається вид зв’язку (з класифікатора) між цими об’єктами.

Для уникнення нелогічних або помилкових зв’язків існують схеми зв’язків (sys_scheme_links, sys_scheme_ap), які містять записи, що вказують між якими реєстрами який зв'язок дійсно можливий. Використовувати ці схеми можна як всередині застосунку, так і на сервері за допомогою тригера, який перед вставкою записів до пов’язуючи таблиць буде перевіряти їх на коректність та у разі необхідності відхиляти додавання незадовільних записів.

Для додання до таблиці необов’язкових даних, або, так названих додаткових «властивостей» (ap — additional property), використовується табличка з назвою ap_типДаних, де типДаних — це найменування типу даних для значення цієї властивості.

Для прикладу можна розглянути обмеження терміну проходження для тесту. Для цього в класифікатор зв’язків додаються зв’язки «Дата початку тесту» та «Дата закінчення тесту», а потім до таблиці ap_datetime вноситься запис, що містить дані про об’єкт, якого ця властивість стосується (реєстр та id), вид зв’язку та саме значення елементарної властивості.

Така схема дозволяє додавати скільки завгодно додаткових даних до об’єктів, не нарощуючи і не засмічуючи при цьому головні таблиці.

Крім цього, до структури БД входить ще одна таблиця — sys_info, що повинна зберігати службову інформацію для застосунку у вигляді назви параметра та його значення. Там міститься, наприклад, пароль, виданий організацією для викладачів і необхідний для увімкнення функцій для викладача у системі (запитується при встановленні ознаки «є викладачем» у редакторі особистих даних, описаного у 3.3.2).

### 4.2.1 Структура таблиць

Як було зазначено вище, у базі даних усього 8 таблиць, що представляють об’єкти, необхідність яких обумовлюється предметною областю та бізнес-логікою програми.

Так, до основних складових БД належать такі таблиці, як:

— «Тест» (test) з колонками, зображеними на рисунку 4.2;

— «Питання» (question) з колонками, зображеними на рисунку 4.3;

— «Відповідь» (answer) з колонками, зображеними на рисунку 4.4;

— «Користувачі» (users) з колонками, зображеними на рисунку 4.5;

— «Особи» (entities) з колонками, зображеними на рисунку 4.6;

— «Навчальні одиниці» (units) з колонками, зображеними на рисунку

4.7;

— «Огляд тесту» (test_survey) з колонками, зображеними на рисунку 4.8;

— «Огляд відповіді» (answer_survey) з колонками, зображеними на рисунку 4.9.

Нижче подані згадані рисунки 4.1 – 4.6, на яких зображено колонки з зазначення для них типу даних, властивостей та обмежень для кожної з перелічених таблиць. Також за наявності зовнішніх ключів у таблиці вони будуть описані під відповідним рисунком.

Щодо зовнішніх ключів варто зауважити, що MySQL дозволяє контролювати таблиці-нащадки під час оновлення або видалення даних в батьківській таблиці за допомогою виразів ON UPDATE і ON DELETE.

Так, MySQL підтримує 5 дій, які можна використовувати у даних виразах:

— CASCADE: якщо пов'язаний запис батьківської таблиці було оновлено або видалено, то відповідні записи в таблицях-нащадках також будуть оновлені або видалені;

— SET NULL: при оновленні або видаленні запису в батьківській таблиці значенням в дочірній таблиці буде присвоєно NULL;

— NO ACTION: дивись RESTRICT;

— RESTRICT: якщо пов'язані записи батьківської таблиці оновлюються або видаляються, то база даних не дозволить змінювати записи в батьківській таблиці.

Обидві команди NO ACTION і RESTRICT еквівалентні відсутності виразів ON UPDATE та ON DELETE для зовнішніх ключів.

| Column Name | Datatype | PK | NN | UQ | B | UN | ZF | AI |
|---|---|---|---|---|---|---|---|---|
| id | INT(11) | ☑ | ☑ | ☐ | ☐ | ☐ | ☐ | ☑ |
| name | VARCHAR(120) | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ |
| idKind | INT(11) | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ |
| idAuthor | INT(11) | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ |

Рисунок 4.2 — Колонки таблиці «Тест»

Відповідно до відношень між об’єктами предметної області, у таблиці «Тест» сформовані такі зовнішні ключі:

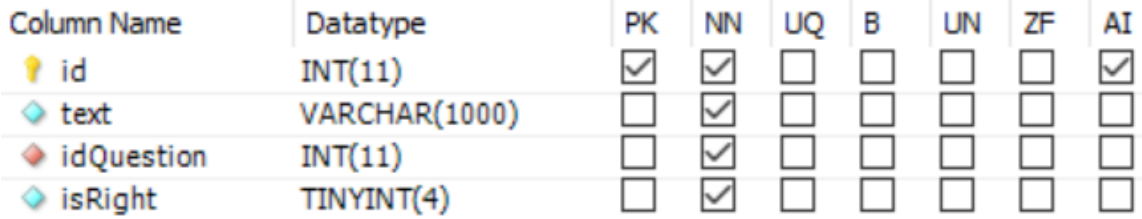
— author_id посилається на id відповідної таблиці, дія на редагування та на видалення — CASCADE;

— kind_id посилається на id запису з системного класифікатору, дія на редагування та на видалення — RESTRICT.

| Column Name | Datatype | PK | NN | UQ | B | UN | ZF | AI | G | Default |
|---|---|---|---|---|---|---|---|---|---|---|
| id | INT(11) | ☑ | ☑ | ☐ | ☐ | ☐ | ☐ | ☑ | ☐ | |
| text | MEDIUMTEXT | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | |
| idTest | INT(11) | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | |
| points | FLOAT | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | '1' |

Рисунок 4.3 — Колонки таблиці «Питання»

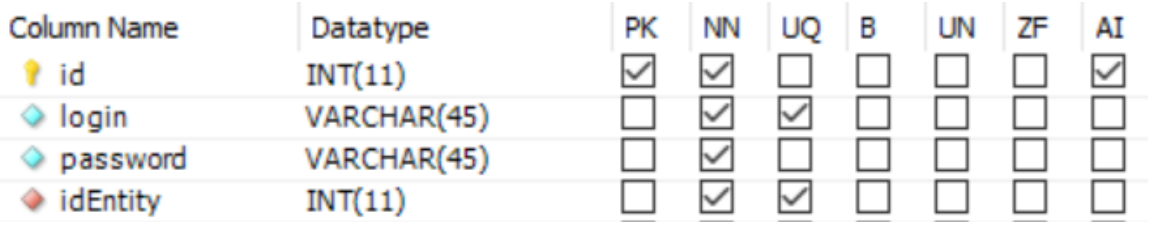
Для таблиці «Питання» був сформований зовнішній ключ test_id, що посилається на id тесту, дія на редагування та на видалення — CASCADE.



| Column Name | Datatype | PK | NN | UQ | B | UN | ZF | AI |
|---|---|---|---|---|---|---|---|---|
| id | INT(11) | ☑ | ☑ | ☐ | ☐ | ☐ | ☐ | ☑ |
| text | VARCHAR(1000) | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ |
| idQuestion | INT(11) | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ |
| isRight | TINYINT(4) | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ |

Рисунок 4.4 — Колонки таблиці «Відповідь»

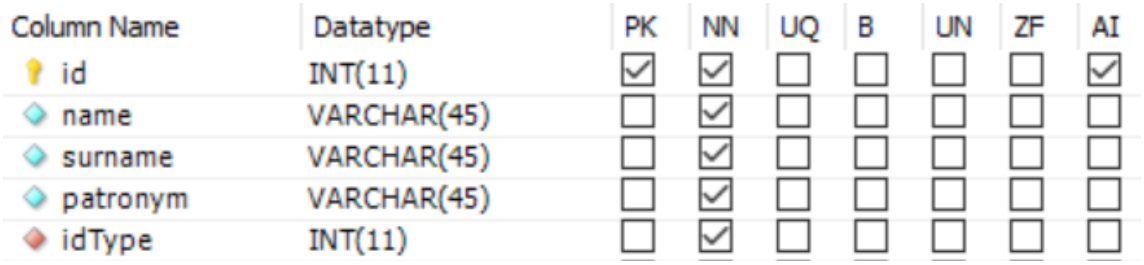
Зображена зверху, на рисунку 4.4, таблиця представляє сутність Відповідь і має зовнішній ключ question_id, що посилається на id питання, варіантом якого є дана відповідь. Для цього ключа була зазначена дія CASCADE на редагування та на видалення.



| Column Name | Datatype | PK | NN | UQ | B | UN | ZF | AI |
|---|---|---|---|---|---|---|---|---|
| id | INT(11) | ☑ | ☑ | ☐ | ☐ | ☐ | ☐ | ☑ |
| login | VARCHAR(45) | ☐ | ☑ | ☑ | ☐ | ☐ | ☐ | ☐ |
| password | VARCHAR(45) | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ |
| idEntity | INT(11) | ☐ | ☑ | ☑ | ☐ | ☐ | ☐ | ☐ |

Рисунок 4.5 — Колонки таблиці «Користувачі»

Для таблиці «Користувачі» відповідно до відношень між об’єктами предметної області був сформований зовнішній ключ entity_id, що посилається на id відповідної таблиці, яка містить інформацію про особу, з якою пов'язаний конкретний обліковий запис. Для цього ключа була зазначена дія RESTRICT на редагування та на видалення.



| Column Name | Datatype | PK | NN | UQ | B | UN | ZF | AI |
|---|---|---|---|---|---|---|---|---|
| id | INT(11) | ☑ | ☑ | ☐ | ☐ | ☐ | ☐ | ☑ |
| name | VARCHAR(45) | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ |
| surname | VARCHAR(45) | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ |
| patronym | VARCHAR(45) | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ |
| idType | INT(11) | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ |

Рисунок 4.6 — Колонки таблиці «Особи»

У таблиці «Особи» є єдиний зовнішній ключ type_id, що посилається на системний класифікатор, дія на редагування та на видалення — RESTRICT.

| Column Name | Datatype | PK | NN | UQ | B | UN | ZF | AI | G | Default |
|---|---|---|---|---|---|---|---|---|---|---|
| id | INT(11) | ☑ | ☑ | ☐ | ☐ | ☐ | ☐ | ☑ | ☐ | |
| name | VARCHAR(250) | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | |
| shortname | VARCHAR(45) | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | NULL |
| idParent | INT(11) | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | NULL |

Рисунок 4.7 — Колонки таблиці «Навчальні одиниці»

Таблиця «Навчальні одиниця» призначена для зберігання ієрархічної структури навчальних одиниць (підрозділів) учбового закладу, наприклад інститутів, факультетів, кафедр, груп тощо. Тому кожен запис, що представляє собою організаційну одиницю, містить посилання на батьківський запис з цієї ж таблиці. Для структурних одиниць, що знаходяться на горі ієрархії значення даного поля — NULL.

Ієрархічні посилання організовані за допомогою зовнішнього ключа unit_parent_id, який пов’язує поле idParent з полем id батьківського запису цієї ж таблиці. Для цього ключа була зазначена дія CASCADE на редагування та на видалення.

| Column Name | Datatype | PK | NN | UQ | B | UN | ZF | AI |
|---|---|---|---|---|---|---|---|---|
| id | INT(11) | ☑ | ☑ | ☐ | ☐ | ☐ | ☐ | ☑ |
| idEntity | INT(11) | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ |
| idTest | INT(11) | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ |
| dateBegin | TIMESTAMP | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ |
| dateEnd | TIMESTAMP | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ |
| points | FLOAT | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ |

Рисунок 4.8 — Колонки таблиці «Огляд тесту»

У таблиці під назвою «Огляд тесту» наявні такі зовнішні ключі:

— author_id посилається на id відповідної таблиці, дія на редагування та на видалення — CASCADE;

— kind_id посилається на id запису з системного класифікатору, дія на редагування та на видалення — RESTRICT.

| Column Name | Datatype | PK | NN | UQ | B | UN | ZF | AI |
|---|---|---|---|---|---|---|---|---|
| id | INT(11) | ☑ | ☑ | ☐ | ☐ | ☐ | ☐ | ☑ |
| idTestSurvey | INT(11) | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ |
| idQuestion | INT(11) | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ |
| idAnswer | INT(11) | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ |

Рисунок 4.9 — Колонки таблиці «Огляд відповіді»

Для організації збереження результатів проходження вихованцями тестів, була створена таблиця «Огляд відповіді». Усі її поля є ідентифікаторами: ідентифікатор запису у даній таблиця, ідентифікатор огляду тесту, в рамках якого була надана ця відповідь, ідентифікатор питання, якого стосується ця відповідь, та ідентифікатор самої відповіді.

Таким чином, усі поля цієї таблиці, окрім id, пов’язані з іншими таблицями, для чого використовуються наступні зовнішні ключі:

— test_survey_id посилається на id відповідної таблиці, дія на редагування та на видалення — CASCADE;
— question_id_survey посилається на id запису з таблиці «Питання», дія на редагування та на видалення — RESTRICT;
— answer_id_survey посилається на id запису з таблиці «Відповідь», дія на редагування та на видалення — RESTRICT.

### 4.2.2 Застосування збережених процедур

Як правило, щоб зменшити навантаження на машину клієнта, частину логіку можна перенести на сервер, де знаходяться готові SQL запити для модифікації або отримання даних відповідно до відомих популярних інформаційних потреб користувачів. Для цього в БД зберігаються процедури [17], які можуть приймати вхідні параметри та здійснювати перетворення або вибір необхідних даних.

Розглянемо приклад такої процедури наведений у додатку Б.

Дана процедура призначена для повернення усіх доступних для проходження певному користувача тестів. На вході вона приймає обов’язковий параметр — ідентифікатор користувача у системі. Цей вхідний параметр дозволяє їй знайти відповідну особу, що представляє даного користувача, та відшукати пов’язані з нею навчальні структурні одиниці.

Варто зауважити, що враховується й те, що тест може бути призначений не кінцевій одиниці (групі), а більш узагальненій (наприклад, факультету), що знаходиться вище по ієрархії. Для цього за допомогою курсора здійснюється прохід по зв’язаним навчальним одиницям і додається інформація про ті одиниці, до складу яких вони належать.

Далі за допомогою цих навчальних одиниць та наявних між таблицями зв’язків здійснюється пошук тестів, які призначені для них. Щоб до вибірки входили лише релевантні записи під час пошуку тестів здійснюється

додаткова фільтрація по обмеженню дати проходження тесту (якщо воно присутнє) відносно поточної дати.

В решті-решт, з отриманої вибірки видаляються тести, що є контрольними та вже були пройдені користувачем, оскільки згідно з логіки предметної області повторне проходження допускається лише для тренувальних тестів, що мають на меті підготовку учня чи студента до відповідального тесту-контрольної.

Отримана таким чином інформація повертається застосунку, який викликав цю процедуру, і відображається у ньому.

З описаної процедури можна прийти до висновку, що для того, щоб знайти доступні користувачу тести для проходження, застосунку прийшлось би зберігати в пам’яті багато об’єктів з безліччю зв’язків. Серед них він би мав здійснювати пошук релевантних тестів, застосовуючи подібний за логікою, але відмінний за реалізацією, алгоритм, що включав би в себе безліч перевірок та фільтрацію. Застосування ж збережених процедур дозволяє уникнути цього.

Переважна більшість інших процедур, що використовуються у даному програмному рішенні, призначена для отримання інформації в межах модуля статистики. Такі процедури, на відміну від розглянутої, повертають розширену інформацію, отриману з декількох таблиць, і зазвичай використовують наявні у MySQL функції. Так, наприклад, в модулі статистики для викладача для підрахунку кількості проходжень тесту використовується функція COUNT.

## 4.3 ВИСНОВКИ

Розділ №4 описує структуру, а також спосіб організації даних у базі даних, розробленій для системи тестування. БД зберігає інформацію необхідну для роботи застосунку й до її складу входять 18 таблиць.

З них 8 представляють моделі предметної області, 3 зберігають додаткові властивості для цих моделей, а решта призначені для зв’язування записів з різних таблиць та збереження службової інформації. Кожна основна таблиця

була детально розглянута з зазначенням її елементів: колонки, первинні та зовнішні ключі.

Також у цьому розділі було надано загальний опис та пояснення щодо застосування збережених процедур, які надають можливість виконувати частину операцій на сервері БД, що зменшує навантаження на клієнта та прискорює роботу системи.

# 5 ОПИС ПРОГРАМИ

## 5.1 ВІДОМОСТІ ПРО ПРОГРАМУ

Дана програма носить назву «Система тестування для навчальних закладів» та відноситься до настільних застосунків, розроблених під операційну систему Microsoft Windows з використання платформи .NET Framework 4. В розробленому програмному рішенні передбачено та протестовано взаємодію клієнтського застосунку з такою СКБД, як MySQL, де зберігається дані.

Для роботи з БД застосунок вимагає встановлення драйверу MySQL Connector/NET, який можна завантажити з офіційного сайту MySQL. ADO.NET є важливою частиною платформи .NET. Це специфікація, яка об'єднує доступ до реляційних баз даних, файлів XML та інших даних застосунків. MySQL Connector/NET — це реалізація специфікації ADO.NET для бази даних MySQL.

Параметри підключення до БД прописані у конфігураційному файлі, що знаходиться поряд з виконуваним файлом та має розширення exe.config і можуть бути змінені користувачем. Для цього необхідно зміни атрибут connectionString однойменного тегу, перебудовувати рішення при цьому не треба.

Також для функціонування системи поряд з виконуваним файлом мають знаходитися наступні взяті з зовнішніх джерел бібліотеки:

— MySql.Data.dll (на даних час використовується остання стабільна версія 6.9.12) — бібліотека класів для . NET клієнтів MySql;

— Xceed.Wpf.Toolkit.dll — безкоштовний інструментарій з відкритим кодом, що надається за ліцензією Microsoft Public License, який містить розширену колекцію графічних елементів для WPF.

Дані бібліотеки постачаються разом з програмним рішенням кінцевому користувачу.

Так, користувачу постачається папка, що містить 4 файли: виконуваний файл для запуску програми, конфігураційний файл та 2 бібліотеки, описані

вище. Розмір виконуваного файлу програми не перевищує 1 МБ, а уся тека займає близько 3 МБ.

В той час сам проект був розроблений за допомогою інтегрованого середовища розробки програмного забезпечення Microsoft Visual Studio та містить 26 тек та 116 файлів, включаючи пакети бібліотек та ресурсів програми: 14 картинок. Програма має власний значок, який також встановлений як іконка вікон у системі. Усі графічні ресурси програми вбудовані у саме рішення.

Логічну структуру програми програми наведено у кресленику ІТ51.12.0БАК.004 Д2, який містить UML-діаграму, що зображує схему класів. Опис архітектури системи та її основних модулів був наведений у попередніх розділах цієї роботи.

## 5.2 АЛГОРИТМИ ОЦІНЮВАННЯ РЕЗУЛЬТАТІВ ТЕСТУ

В системі передбачено два види тестів: контрольні та тренувальні, а також два види питань: з фіксованою кількістю відповідей та з зазначенням усіх можливих відповідей. Алгоритм оцінки відповідей не залежить від типу тесту, а відрізняється лише залежно від типу питання.

Варто нагадати, що для кожного питання користувач може зазначити кількість балів та правильних відповідей. Таким чином, бал за одну правильну відповідь (БОВ) для питання обчислюється як загальна кількість балів поділена на кількість правильних відповідей в рамках цього питання.

Отож, для питань, які не мають ознаки «Оберіть усі правильні відповіді» алгоритм підрахунку балів є досить простим: за кожну правильну відповідь користувач отримує БОВ. Оскільки системи обмежує вибір кількості відповідей для такого типу питань, то максимальна кількість, яку може набрати користувач дорівнює кількості балів за питання, а мінімальна становить 0.

Такий алгоритм підрахунку не підходить для питань, де необхідно зазначити усі можливі правильні відповіді. Це пов’язано з тим, що вихованець може просто позначити усі варіанти відповідей як правильні, і тоді згідно з

алгоритмом, описаним в попередньому абзаці, система нарахуємо йому максимальний бал.

Тому для уникнення такого шахрайства в алгоритм обробки відповідей внесені певні зміни: для питань з вибором усіх правильних відповідей за кожне вибране правильне питання користувач отримує БОВ, а за неправильне втрачає. Якщо в кінці сума балів за питання — негативна, то за дане питання користувача нараховується 0 балів, таким чином, щоб вкінці оцінювання тесту мінімальна оцінка не була нижчою за 0.

Блок-схема описаного алгоритму оцінки результатів тесту представлена на рисунку 5.1.

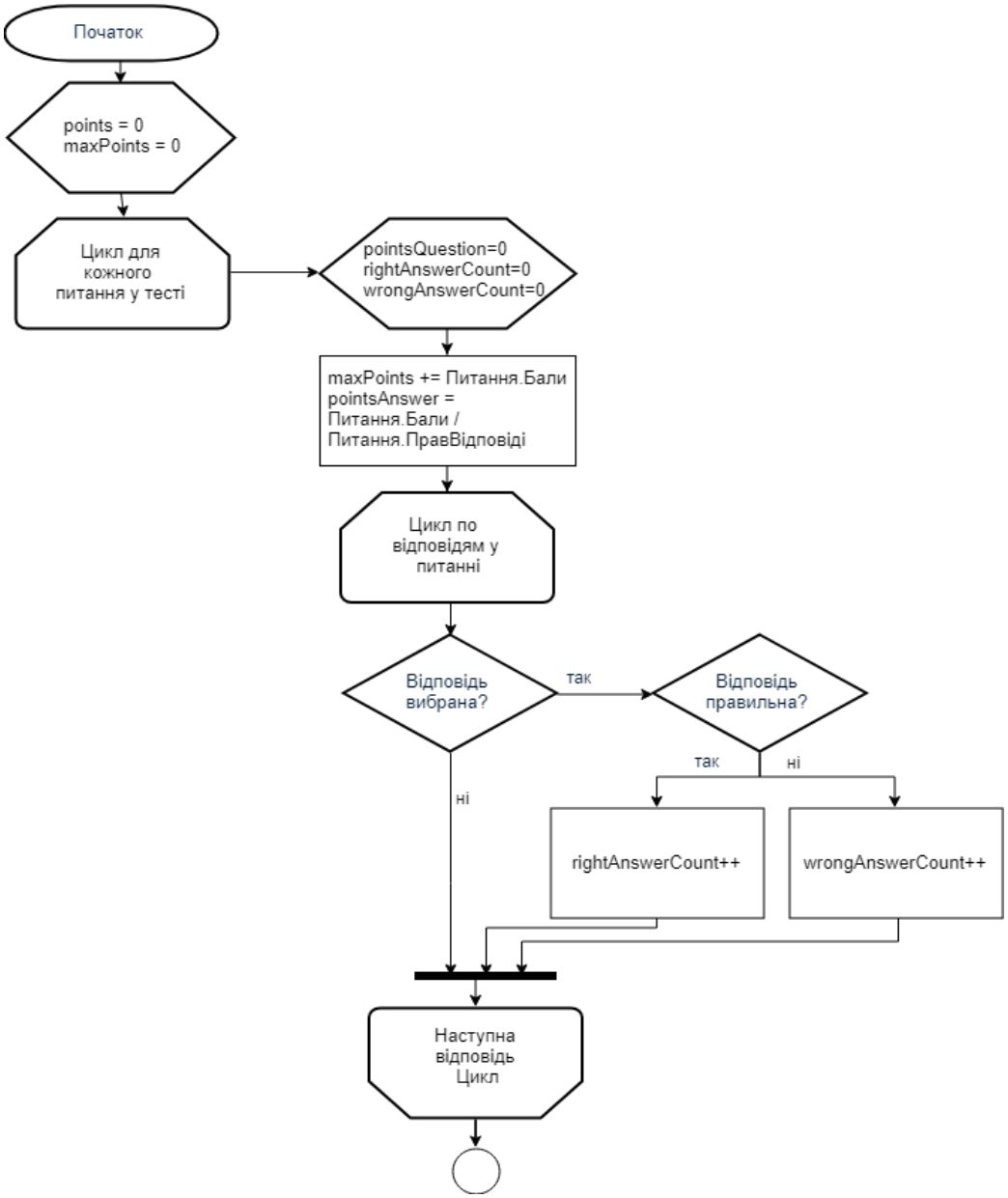


Рисунок 5.1 — Блок-схема алгоритму оцінки результатів тесту

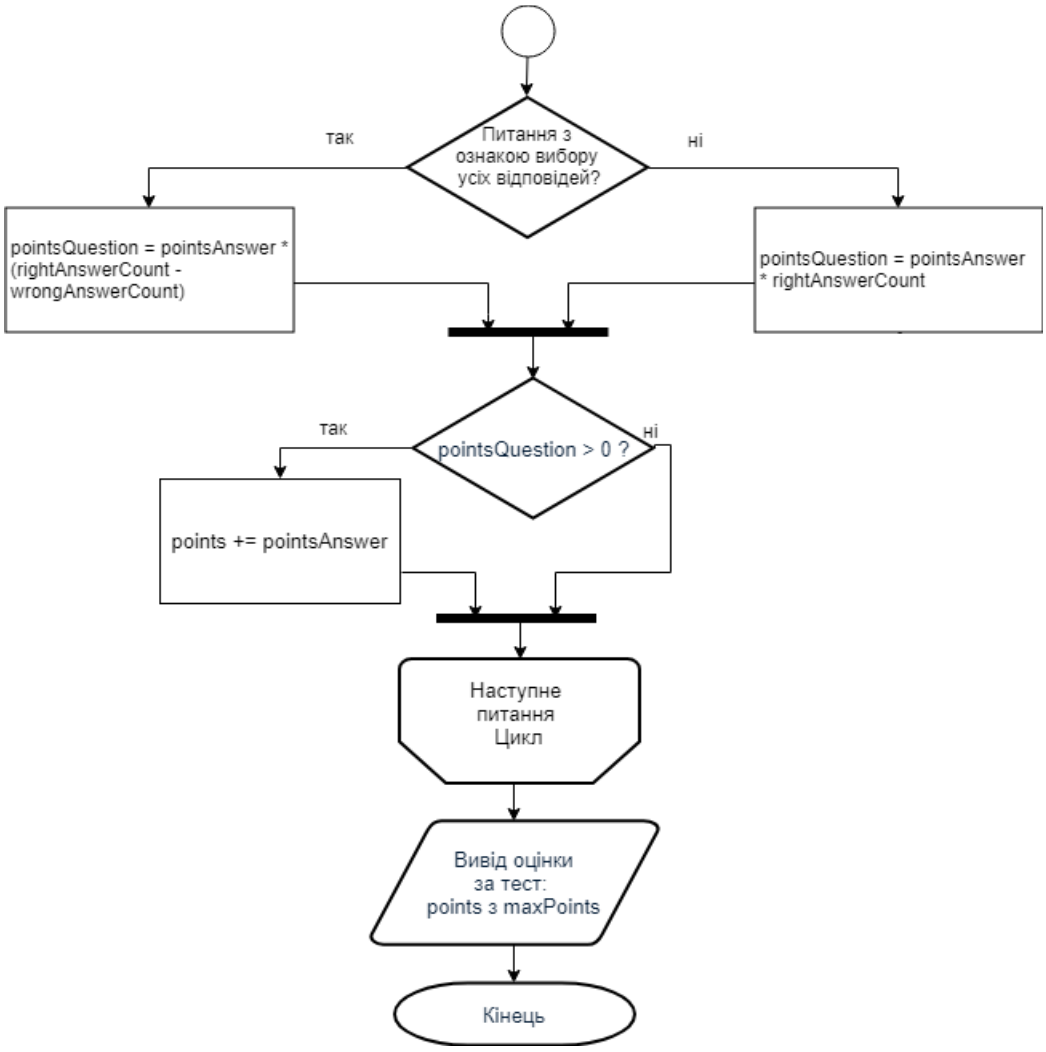



Рисунок 5.1, аркуш 2

## 5.3 ВИСНОВКИ

Отже, у даному розділі був наданий стислий опис системи та необхідних для її роботи компонентів: самого виконуваного файлу програми, сторонніх бібліотек і конектора MySQL. Також було зазначено кількість тек та файлів в межах програмного рішення та обсяг пам’яті, яку вони займають.

В другому підрозділі було пояснено принцип оцінювання результатів проходження тесту. Підсумовуючи, можна сказати, що для питань, де необхідно обрати фіксовану кількість відповідей, алгоритм нараховує бали за кожну правильну відповідь, а для питань з опцією «Оберіть усі правильні

відповіді» він також віднімає бали за неправильні варіанти, що не дозволяє обдурити систему просто обравши усі варіанти відповідей .

# 6 СЦЕНАРІЇ ВИКОРИСТАННЯ ПРОГРАМИ

## 6.1 ДІАГРАМА ПРЕЦЕДЕНТІВ

Діаграма прецедентів представлена у вигляді графа, що складається з множини акторів (передбачені ролі користувачів у системі) та прецедентів (варіантів використання) обмежених прямокутником системи. Усі об’єкти на діаграмі пов’язані між собою за допомогою асоціацій між акторами та прецедентами, відношень серед прецедентів та відношень узагальнення між акторами [18, 19]. Діаграми прецедентів відображають елементи моделі варіантів використання.

Розроблена система представляється у вигляді безлічі сутностей, чи акторів, які взаємодіють із системою за допомогою так званих варіантів використання — описання послуг, які система надає користувачу.

Іншими словами, кожен варіант використання є функціональною можливістю та визначає деякий набір дій, який виконує система при діалозі з певним актором. Інформація про те, яким саме чином буде реалізована взаємодія акторів з системою навмисно приховується.

Діаграма прецедентів для розробленого програмного рішення наведена у кресленику ІТ51.12. 0БАК.003 Д1. Як можна побачити, на ній зображено 2 актора, що відповідають передбаченим типам користувачів: викладач та вихованець, а також 14 варіантів використання, деякі з яких є розширеннями або включення інших прецедентів.

Для опису кожного з варіантів використання побудуємо таблицю із наступними полями:

— назва діяльності;

— унікальний ідентифікатор;

— короткий опис;

— тригер;

— попередні умови для виконання;

— умови після виконання;

— основний та альтернативний потоки розвитку;

— можливі помилки.

Перший варіант використання — «Вхід до системи», що включає в себе варіант використання «Авторизація». Опис діяльності для цього варіанту використання представлений у таблиці 6.1. Варто зауважити, що основним потоком для цієї діяльності є вхід до системи в офлайн режимі, оскільки авторизація виділена як окрема діяльність.

Таблиця 6.1 — Опис діяльності «Вхід до системи»

| | |
|---|---|
| Назва | Вхід до системи |
| ID | C_01 |
| Опис | Одразу після запуску клієнтської програми, користувачу відкривається вікно, де він може здійснити вхід до системи двома способами: завдяки авторизації у системі під своїм логіном та паролем чи натиснувши ЛКМ на мітці «Оффлайн режим» |
| Актор | Викладач, вихованець |
| Тригер | Запуск виконуваного файлу |
| Післяумови | Користувач здійснив вхід до системи, йому відобразилося головне меню програми |
| Основний потік розвитку | 1. Користувач натиснув ЛКМ по мітці для офлайн роботи.<br>2. З’явилося головне меню, що містить пункт для проходження тестів. |
| Альтернативний потік розвитку | 1. Користувач увійшов до системи за допомогою модуля авторизації.<br>2. З’явилося головне меню, що містить більшу кількість доступних пункт. |

| | |
|---|---|
| Можливі помилки | Відсутні для входу до системи в офлайн режимі |

Таблиця 6.2 містить інформацію про діяльність, яка відповідає варіанту використання «Авторизація». Як вже зазначалося, цей варіант використання є частиною варіанту використання «Вхід до системи».

Таблиця 6.2 — Опис діяльності «Авторизація»

| | |
|---|---|
| Назва | Авторизація |
| ID | C_02 |
| Опис | Одразу після запуску клієнтської програми, користувачу відкривається вікно, де він може здійснити вхід до системи під своїм обліковим записом за своїм логіна та пароля |
| Актор | Викладач, вихованець |
| Тригер | Запуск виконуваного файлу |
| Післяумови | Користувач здійснив вхід до системи від свого імені, йому відобразилося головне меню програми, що містить релевантну для даної особи інформацію |
| Основний потік розвитку | 1. Користувач ввів правильні логін та пароль.<br>2. Натиснув кнопку «Увійти».<br>3. З’явилося головне меню, що містить відповідні для його ролі пункти меню. |
| Альтернативний потік розвитку | 1. Користувач ввів некоректні дані.<br>2. Натиснув кнопку «Увійти».<br>3. З’явилося повідомлення від системи про відсутність облікового запису з таким логіном або про невірно введений пароль. |
| Можливі помилки | Відсутність зв’язку з БД, що унеможливлює перевірку даних облікового запису |

Далі у таблиці 6.3 наведено опис такого варіанту використання, як «Редагування особистих даних». Цей варіант використання, в свою чергу включає в себе такий варіант використання, як «Редагування облікового запису», оскільки вони обидва реалізовані за допомогою редактора особистих даних, що містить інформацію і про особу і про її обліковий запис.

Таблиця 6.3 — Опис діяльності «Редагування особистих даних»

| | |
|---|---|
| Назва | Редагування особистих даних |
| ID | C_03 |
| Опис | Авторизувавшись у системі, користувач у стартовому меню обрав пункт «Особисті дані», після чого йому відкрився редактор особистих даних |
| Актор | Викладач, вихованець |
| Тригер | Натискання ЛКМ по пункту меню «Особисті дані» |
| Післяумови | Внесені користувачем зміни в даних були застосовані до об’єктів та збережені до БД |
| Основний потік розвитку | 1. Користувач змінює дані на вкладці «Особисті дані» та/або «Навчальний заклад».<br>2. Натискає кнопку «Зберегти» або підтверджує бажання внесення змін за допомогою діалогового вікна.<br>3. Редактор особистих даних закривається.<br>4. Користувач опиняється у стартовому меню. |
| Альтернативний потік розвитку | 1. Користувач не ініціював збереження даних у редакторі.<br>2. Користувач закрив редактор.<br>3. Користувач опиняється у стартовому меню. |
| Можливі помилки | Непередбачувані виключення під час збереження даних у базу, пов’язані з втратою зв’язку |

Як вже було зазначено, варіант використання «Редагування облікового запису» здійснюється в рамках того самого модуля, що і попередній розглянутий варіант. Тому характеристика цієї діяльності, представлена у таблиці 6.4, дуже схожа до характеристики суміжної діяльності «Редагування особистих даних», представленої у таблиці 6.3.

Таблиця 6.4 — Опис діяльності «Редагування облікового запису»

| | |
|---|---|
| Назва | Редагування облікового запису |
| ID | C_04 |
| Опис | Повторює опис, наведений для діяльності «Редагування особистих даних» |
| Актор | Викладач, вихованець |
| Тригер | Натискання ЛКМ по пункту меню «Особисті дані» |
| Післяумови | Внесені користувачем зміни в даних були застосовані до об’єктів та збережені до БД |
| Основний потік розвитку | 1. Користувач змінює пароль на вкладці «Обліковий запис».<br>2. Підтверджує свою дію та закриває редактор.<br>3. Користувач опиняється у стартовому меню. |
| Альтернативний потік розвитку | 1. Користувач ініціює зміну пароля на вкладці «Обліковий запис».<br>2. Скасовує (не підтверджує) свою дію та закриває редактор.<br>3. Користувач опиняється у стартовому меню. |
| Можливі помилки | Непередбачувані виключення під час збереження даних у базу, пов’язані з втратою зв’язку |

Наступним варіантом використання, представленим на діаграмі прецедентів є «Створення тесту». Варто зазначити, що на відміну від

попередніх діяльностей, до цієї діяльності мають доступ лише викладачі. Діяльність «Створення тесту» проілюстровано у таблиці 6.5.

Таблиця 6.5 — Опис діяльності «Створення тесту»

| | |
|---|---|
| Назва | Створення тесту |
| ID | C_05 |
| Опис | Авторизувавшись у системі, користувач у стартовому меню обрав пункт «Створити тест», після чого йому відкрився редактор тесту для новоствореного об’єкту |
| Актор | Викладач |
| Тригер | Натискання ЛКМ по пункту стартового меню «Створити тест» |
| Післяумови | До БД додається запис про новий об’єкт з усіма внесеними змінами. Даний тест автоматично пов’язується з особою, якій належить поточний обліковий запис |
| Основний потік розвитку | 1. Користувач зазначає необхідні дані та налаштування для тесту на вкладці «Загальне» та переходить до питань за допомогою однойменної кнопки.<br>2. Користувач додає необхідну кількість питань та варіантів відповідей, а також визначає для них необхідні дані.<br>3. Натискає ЛКМ на кнопці «Зберегти» та виходить з редактору тесту.<br>4. Перед користувачем з’являється стартове меню. |
| Альтернативний потік розвитку | 1. Користувач не ініціював збереження даних у редакторі та закрив його. |

| | |
|---|---|
| Назва | Створення тесту |
| | 2. Користувач опиняється у стартовому меню. |
| Можливі помилки | Непередбачувані виключення під час збереження даних у базу, пов’язані з втратою зв’язку |

У таблиці 6.6 надано пояснення такого варіанту використання, як «Редагування тесту».

Для реалізації цієї та попередньої діяльностей застосовується один й той самий модуль, що називається редактором тесту (дивись розділ 3.3.3). Оскільки потік розвитку, що стосується роботи з даними у редакторі, для цих двох діяльностей незначно відрізняється, то немає необхідності детально його зазначати у таблиці.

Таблиця 6.6 — Опис діяльності «Редагування тесту»

| | |
|---|---|
| Назва | Редагування тесту |
| ID | C_06 |
| Опис | Авторизувавшись у системі, користувач у стартовому меню обрав пункт «Редагувати тест», після чого у списку доступних йому тестів поряд з потрібним тестом натиснув на відповідну піктограму для відкриття редактору |
| Актор | Викладач |
| Тригер | Виклик редактора тесту з вікна, що відображає список створених користувачем тестів доступних для редагування |
| Післяумови | Об’єкт тесту синхронізується з вихідним об’єктом і до БД надсилаються відповідні запити на змінення даних щодо тесту та його обмежень, питань та відповідей. |

| Назва | Редагування тесту |
|---|---|
| Основний потік розвитку | 1. Користувач вносить зміни до існуючого тесту.<br>2. Користувач натискає ЛКМ на кнопці «Зберегти» та виходить з редактору тесту.<br>3. Перед користувачем знову з’являється список доступних для редагування тестів. |

Продовження таблиці 6.6

| Альтернативний потік розвитку | 1. Користувач не ініціював збереження даних у редакторі та закрив редактор.<br>3. Перед користувачем знову з’являється список доступних для редагування тестів. |
|---|---|
| Можливі помилки | Непередбачувані виключення під час збереження даних у базу, пов’язані з втратою зв’язку |

Діяльність «Видалення тесту» детально розглянуто у таблиці 6.7.

Таблиця 6.7 — Опис діяльності «Видалення тесту»

| Назва | Видалення тесту |
|---|---|
| ID | C_07 |
| Опис | Авторизувавшись у системі, користувач у стартовому меню обрав пункт «Редагувати тест», після чого у списку доступних йому тестів поряд з потрібним тестом натиснув на піктограму для його видалення |
| Актор | Викладач |
| Тригер | Натискання піктограми «Видалити» для елементу списку, що формується з пункту меню «Редагувати тест» |

| | |
|---|---|
| Назва | Видалення тесту |
| Післяумови | Тест з усіма пов’язаним з ним дочірніми об’єктами видаляється з БД та системи |
| Основний потік розвитку | 1. Користувач обирає тест зі списку.<br>2. Користувач натискає ЛКМ по відповідній піктограмі поряд з тестом.<br>3. Перед користувачем знову з’являється список доступних для редагування чи видалення тестів. |

Продовження таблиці 6.7

| | |
|---|---|
| Можливі помилки | Непередбачувані виключення під час видалення даних у базі, пов’язані з втратою зв’язку |

Однією з вимог до дипломного проекту була можливість роботи з тренувальними тестами у програмі в умовах відсутності зв’язку з сервером БД. Для цього у застосунку передбачено офлайн режим, де є можливість проходження тренувальних тестів, які зберігаються у файлах на носії користувача. Для цієї мети у системі передбачено дві функції: експорт (вивантаження) тесту на носій та імпорт (завантаження) тесту до системи.

Так, для формування файлу з тренувальним тестом користувач користується діяльністю «Експорт тесту на носій», описаною в таблиці 6.8.

Таблиця 6.8 — Опис діяльності «Експорт тесту на носій»

| | |
|---|---|
| Назва | Експорт тесту на носій |
| ID | C_08 |
| Опис | В умовах доступу до БД, знаходячись у стартовому меню програми, користувач обрав пункт «Проходження тесту» внаслідок чого з’явилося вікно зі списком доступних для проходження та |

| | |
|---|---|
| | вивантаження тестів. В даному списку користувач натискає на піктограму вивантаження тесту на носій, що знаходиться біля обраного тесту |
| Актор | Вихованець |
| Тригер | Виклик функції експорту тесту зі спеціального списку у модулі проходження тесту |
| Післяумови | У вибране користувачем місце на електронному носії буде збережений тест у форматі XML з додатковим шифрування змісту цього файлу |

Продовження таблиці 6.8

| | |
|---|---|
| Основний потік розвитку | 1. Користувач обрав теку або диск, до якої бажає зберегти файл.<br>2. Користувач отримав повідомлення про успішне збереження файлу в обране розташування на носії. |
| Альтернативний потік розвитку | 1. Користувач натиснув на кнопку вивантаження тесту на носій.<br>2. Користувач отримав повідомлення про помилку під час спроби виконання операції. |
| Можливі помилки | Втрата зв’язку з базою даних, проблеми з доступом до файлів на обраному носії |

Діяльність «Імпорт тесту з носія» використовується для завантаження раніше сформованого файлу до системи та здійснюється тим самим модулем, що і вивантаження, але не потребує зв’язку з базою даних. Вона описана у таблиці 6.9.

Таблиця 6.9 — Опис діяльності «Імпорт тесту з носія»

| | |
|---|---|
| Назва | Імпорт тесту з носія |

| | |
|---|---|
| ID | C_09 |
| Опис | У стартовому меню користувач обирає пункт «Проходження тесту» внаслідок чого з’явилося вікно внизу якого він натискає кнопку «Додати з носія» для завантаження до програми тесту, що знаходиться у спеціально підготовленому файлі |
| Актор | Вихованець |
| Тригер | Виклик функції імпорту тесту зі спеціального вікна у модулі проходження тесту |

Продовження таблиці 6.9

| | |
|---|---|
| Післяумови | З вибраного користувачем файлу на електронному носії система внаслідок розшифрування даних та розбору XML сформує об’єкт тесту та додасть його до списку тестів доступних для проходження |
| Основний потік розвитку | 1. Користувач зазначив сформований раніше системою XML-пакет, який бажає завантажити до системи.<br>2. Тест з’явився у списку тестів доступних для проходження. |
| Альтернативний потік розвитку | 1. Користувач натиснув на кнопку завантаження тесту з носія.<br>2. Для імпорту зазначив невірний файл.<br>3. Система повідомила користувача про невдачу імпорту тесту з даного файла. |
| Можливі помилки | Зазначення користувачем невірного файлу або помилка доступу до файлу |

Діяльність «Проходження тесту» визначена у таблиці 6.10.

Таблиця 6.10 — Опис діяльності «Проходження тесту»

| Назва | Проходження тесту |
|---|---|
| ID | C_10 |
| Опис | У стартовому меню користувач обирає пункт «Проходження тесту» внаслідок чого з'явилося вікно зі списком доступних для проходження тестів. В даному списку користувач натискає на піктограму «Почати тестування» внаслідок чого відкривається вікно для проходження тесту |

Продовження таблиці 6.10

| Актор | Вихованець |
|---|---|
| Тригер | Виклик модуля проходження тесту |
| Післяумови | По завершенню тестування система повідомляє користувача про отриману оцінку за тест. При авторизованому вході у систему також буде збережена інформація про результати проходження |
| Основний потік розвитку | 1. Користувач дав відповіді на усі питання та завершив проходження тесту.<br>2. Отримав повідомлення про отриману оцінку.<br>3. Користувач опиняється перед списком тестів для проходження. |

| | |
|---|---|
| Альтернативний потік розвитку | Можливо два шляхи альтернативного розвитку.<br>Перший:<br>1. Користувач не встиг дати відповіді на усі питання та проініціювати завершення тесту, оскільки час проходження, встановлений вихованцем для контрольного тесту, збіг.<br>2. Система припиняє проходження тесту та повідомляє користувачу оцінку, яку він заробив.<br>Другий:<br>1. Користувач дав відповіді не на всі питання, проте проініціював закінчення тестування.<br>2. Система повідомляє користувача про незакінчені питання та активує відповідну вкладку на вікні. |
| Можливі помилки | Помилка збереження результатів до БД у разі втрати зв’язку з нею |

«Перевірка відповідей» — діяльність доступна у межах діяльності «Проходження тесту» для тренувальних тестів та описана у таблиці 6.11.

Таблиця 6.11 — Опис діяльності «Перевірка відповідей»

| | |
|---|---|
| Назва | Перевірка відповідей |
| ID | C_11 |
| Опис | Знаходячись у модулі проходження тестів, користувач натискає ЛКМ на кнопку «Перевірити» |
| Актор | Вихованець |
| Тригер | Натискання кнопки «Перевірити» у модулі проходження тестів |
| Післяумови | Для активного питання буде показано, які з обраних користувачем відповідей, є правильними, а які ні |

| | |
|---|---|
| Основний потік розвитку | 1. Під час проходження тесту користувач натиснув кнопку перевірки.<br>2. Модуль вказав користувачу на його правильні та неправильні відповіді. |
| Можливі помилки | Відсутні |

Зазначені нижче варіанти використання відносяться до модулю статистики та в основному розподілені між акторами.

Так, «Перегляд графіків успішності по тренувальним тестам» призначений для вихованця. Відповідна діяльність описана у таблиці 6.11.

Таблиця 6.12 — Опис діяльності «Перегляд графіків успішності»

| | |
|---|---|
| Назва | Перегляд графіків успішності |
| ID | C_12 |
| Опис | Після авторизації, користувач у стартовому меню обрав пункт «Статистика», після чого у списку пройдених ним тестів поряд з потрібною назвою тренувального тесту натиснув на відповідну піктограму для перегляду графіку успішності |

Продовження таблиці 6.12

| | |
|---|---|
| Актор | Вихованець |
| Тригер | Натискання піктограми «Графік успішності» для елементу списку у модулі статистики |
| Післяумови | Для обраного користувачем тесту відобразиться графік успішності у вигляді ламаної, що співставляє дати проходження тесту до їх результатів у порядку зростання дати |

| | |
|---|---|
| Основний потік розвитку | 1. Серед списку пройдених тестів користувач обирає тренувальний тест.<br>2. Користувач натискає поряд піктограму для перегляду графіку успішності.<br>3. На екрані з’являється графік залежності оцінки за тест від дати його проходження. |
| Можливі помилки | Непередбачувані виключення під час отримання даних з бази, пов’язані з втратою зв’язку |

Такий варіант використання, як «Перегляд оцінок за контрольні тести» доступний для обох акторів. Оскільки для викладачів він є складовою діяльності «Перегляд відповідей на питання», то цей варіант використання буде розглянутий нижче у таблиці 6.13 для актора Вихованець.

Таблиця 6.13 — Опис діяльності «Перегляд оцінок за контрольні тести»

| | |
|---|---|
| Назва | Перегляд оцінок за контрольні тести |
| ID | C_13 |
| Опис | Після авторизації, користувач у стартовому меню обрав пункт «Статистика» |
| Актор | Вихованець |
| Тригер | Відкриття пункту меню «Статистика» |

Продовження таблиці 6.13

| | |
|---|---|
| Післяумови | Відображення списку пройдених користувачем тестів з останньою отриманою оцінкою |
| Основний потік розвитку | 1. Вихованець перейшов до модулю статистики.<br>2. У сформованому списку він подивився оцінку за пройдені тести. |

| | |
|---|---|
| Можливі помилки | Непередбачувані виключення під час отримання даних з бази, пов’язані з втратою зв’язку |

«Перегляд відповідей на питання» — діяльність, що розширює діяльність «Перегляд оцінок за контрольні тести» для викладачів. Її опис, включаючи опис дочірньої діяльності, надано у таблиці 6.14.

Таблиця 6.14 — Опис діяльності «Перегляд відповідей на питання»

| | |
|---|---|
| Назва | Перегляд відповідей на питання |
| ID | C_14 |
| Опис | Після авторизації, користувач у стартовому меню обрав пункт «Статистика», після чого у списку створених ним тестів обрав контрольний тест, що його цікавить, та двічі натиснув ЛКМ по ньому |
| Актор | Викладач |
| Тригер | Подвійне натискання ЛКМ по тесту зі списку, сформованому при відкритті пункту «Статистика» |
| Післяумови | Для обраного користувачем тесту формується список оцінок усіх осіб, що пройшли цей тест (дочірня діяльність «Перегляд оцінок за контрольні тести»).<br>Обравши потрібну особу користувач натискає піктограму для перегляду відповідей цієї особи. |

Продовження таблиці 6.14

| | |
|---|---|
| Основний потік розвитку | Пункти 1-2 описують також основний потік розвитку для діяльності «Перегляд оцінок за контрольні тести». |

| | |
|---|---|
| | 1. У початкову списку своїх тестів користувач двічі натиснув на контрольний тест, що його цікавить.<br>2. Для цього тесту з’явився список з результатами проходжень іншими користувачами.<br>3. Користувач обирає особу для перегляду її відповідей та натискає відповідну піктограму.<br>4. Користувачу відображається вікно, що містить детальну інформацію про відповіді обраної особи. |
| Альтернативний потік розвитку | 1. У початкову списку своїх тестів користувач двічі натиснув на тренувальний тест.<br>2. Реакція системи не відбулася. |
| Можливі помилки | Непередбачувані виключення під час отримання даних з бази, пов’язані з втратою зв’язку |

Останній варіант використання — Перегляд кількісної статистики — дозволяє подивитися загальну кількість проходжень по тесту та кількість проходжень унікальними користувачами для обох видів тесту. Дана діяльність представлена у таблиці 6.15 і призначена виключно викладачам, оскільки тільки їх ролі доступне створення тестів.

Таблиця 6.15 — Опис діяльності «Перегляд кількісної статистики»

| | |
|---|---|
| Назва | Перегляд кількісної статистики |
| ID | C_15 |
| Опис | Після авторизації, користувач у стартовому меню обрав пункт «Статистика» |

Продовження таблиці 6.15

| | |
|---|---|
| Актор | Викладач |
| Тригер | Відкриття пункту меню «Статистика» |

| | |
|---|---|
| Післяумови | Відображення списку створених користувачем тестів з зазначенням:<br>– загальної кількості проходжень усіма користувача<br>– кількості проходжень унікальними користувачами |
| Основний потік розвитку | 1. Викладач перейшов до модулю статистики.<br>2. У сформованому списку він подивився кількісну статистику проходжень для створеним ним тестів. |
| Можливі помилки | Непередбачувані виключення під час отримання даних з бази, пов’язані з втратою зв’язку |

На основі описаних варіантів використання та сформованих у розділі 1.1 функціональних вимог було побудовано таблицю зв’язків між ними — таблиця 6.16. У даній таблиці FR — функціональна вимога, а UC — варіант використання.

Таблиця 6.16 — Requirements Traceability Matrix для варіантів використання

| UC / FR | C_01 | C_02 | C_03 | C_04 | C_05 | C_06 | C_07 | C_08 | C_09 | C_10 | C_11 | C_12 | C_13 | C_14 | C_15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Авторизація | X | X | | | | | | | | | | | | | |
| Редагування особистих даних | | | X | X | | | | | | | | | | | |
| Створення тестів | | | | | X | | | | | | | | | | |
| Редагування тестів | | | | | | X | | | | | | | | | |

Продовження таблиці 6.16

| UC / FR | C_01 | C_02 | C_03 | C_04 | C_05 | C_06 | C_07 | C_08 | C_09 | C_10 | C_11 | C_12 | C_13 | C_14 | C_15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Видалення тестів | | | | | | | X | | | | | | | | |
| Проходження офлайн | | | | | | | | X | X | | | | | | |
| Проходження тестів | | | | | | | | | | X | | | | | |
| Перевірка відповідей | | | | | | | | | | | X | | | | |
| Перегляд успішності | | | | | | | | | | | | X | X | X | |
| Перегляд статистики | | | | | | | | | | | | | | | X |

З таблиці 6.16 можна побачити, що кожна функціональна вимога покрита принаймні одним варіантом використання, що свідчить про функціональну завершеність розробленого програмного продукту. А відсутність варіантів використання, яким не відповідає жодна функціональна вимога, говорить про те, що розроблена система не містить незатребуваних у завданні функцій.

## 6.2 ВИСНОВКИ

Отже, у даному розділі було детально описано діаграму прецедентів системи тестування. Так, на ній визначено два актори (викладач та вихованець) та доступні їм варіанти використання — прецеденти. Таких прецедентів передбачено 14, з яких 4 є спільними для обох акторів.

Для кожного варіанту використання було надано опис діяльності користувача у системі та в кінці розділу побудовано таблицю, що надає інформацію щодо покриття функціональних вимог до системи реалізованими у ній варіантами використання. Таким чином, можна зробити висновок, що розроблена система повністю задовольняє поставленим до неї вимогам.

# ВИСНОВКИ

У ході виконання даного дипломного проекту було розроблену систему тестування для навчальних закладів, яка має забезпечувати освітні заклади засобом проведення контролю знань своїх вихованців у вигляді тестів та підготовки до цих контрольних тестів.

Програма логічно поділена на шість модулів:

— модуль входу до системи, що передбачає авторизований вхід та вхід без авторизації для роботи без необхідності підключення до БД (офлайн);

— модуль редагування особистих даних користувача, що включає можливість зміни паролю до облікового запису;

— модуль управління тестами, який надає можливість створити, змінити та видалити тести;

— модуль експорту та імпорту тестів на носій для роботи офлайн;

— модуль проходження тестів;

— модуль статистики.

Програма призначена для ОС Windows з мінімальною версією платформи .NET Framework 4. Вона написана за допомогою об’єктно-орієнтованої мови високого рівня C#, з використанням технології WPF та застосуванням шаблону проектування MVVM. В якості СКБД, з якою взаємодіє клієнтський застосунок, було обрано реляційну MySQL. Спроектована база даних містить основні та допоміжні таблиці, а також збережені процедури.

# СПИСОК ЛІТЕРАТУРИ


1. Вигерс К. Разработка требований к программному обеспечению: пер с англ. / Вигерс Карл, Битти Джой — М.: Русская Редакция, 2004. — 314 с.
2. Desktop Operating System Market Share Worldwide [Електронний ресурс]/ StatCounter — Режим доступу: http://gs.statcounter.com/os-market-share/desktop.
3. .NET Framework 4 [Електронний ресурс]: відомості та системні вимоги — Режим доступу: https://www.microsoft.com/ru-ru/download/details.aspx?id=17718.
4. Шилдт Г. Полный справочник по C#.: пер. с англ. / Герберт Шилдт: — М. Вильямс, 2004. — 752 с.
5. Метью Мак-Дональд. WPF: Windows Presentation Foundation в .NET 4.0 с примерами на C# 2010 для профессионалов. — М.: Вильямс, 2011. — 1024 с.
6. Андерсон К. Основы Windows Presentation Foundation: пер. с англ / Крис Андерсон. — СПб.: БХВ-Петербург, 2008. — 432 с.
7. Адамс Д. DirectX: Продвинутая анимация: пер. с англ. / Джим Адамс — М.: КУДИЦ-ПРЕСС, 2004. — 480 с.
8. Мейер Д. Теория реляционных баз данных: пер. с англ. / Давид Мейер — М.: Мир, 1987. – 608 с.
9. Васвани В. MySQL: Использование и администрирование MySQL: пер. с англ. / Викрам Васвани — М.: «Питер», 2011. — 368 с.
10. Чкалов А. П. Базы данных: от проектирования до разработки приложений / А. П. Чкалов — СПб.: БХВ-Петербург, 2003. – 384 с.
11. MySQL 8.0 Reference Manual [Електронний ресурс]: Connectors and APIs/ MySQL Engineering Blogs — Режим доступу: https://dev.mysql.com/doc/refman/8.0/en/connectors-apis.html.

12. Кириллов В. В. Структуризованный язык запросов (SQL): учебн. пособ.: [Електронный ресурс]/ В. В. Кириллов, Г. Ю. Громов. – СПб: Санкт-Петерб. госуд. техн. универ., каф. выч. техники, 1998. – Режим доступа к пособию: http://www.citforum.ru/database/sql_kg/.
13. Introduction to Model/View/ViewModel pattern for building WPF apps [Електронний ресурс]/ Microsoft Developer Network. Microsoft — Режим доступу: https://blogs.msdn.microsoft.com/johngossman/2005/10/08/introduction-to-modelviewviewmodel-pattern-for-building-wpf-apps/.
14. Riverst R. The MD5 Message-Digest Algorithm/ Ronald Riverst // RFC — Task Force, 1992. — 21 с.
15. Панасенко С.П. Алгоритмы шифрования. Специальный справочник/ С.П. Панасенко — СПб.: Издательский дом «БХВ-Петербург», 2009 — 576 с.
16. Баричев С.Г. Стандарт AES. Алгоритм Rijdael: Основы современной криптографии/ С.Г. Баричев, В.В. Гончаров, Р. Е. Серов — 3-е изд. — М.: Диалог-МИФИ, 2011.— 176 с.
17. Харрингтон Дж. Л. Проектирование реляционных баз данных: пер. с англ. / Джен Харрингтон Дж. Л. — М.: Лори, 2006 —232 с.
18. Ларман К. Применение UML 2.0 и шаблонов проектирования: пер. с англ. / Крэг Ларман. — 3-е изд. М.: Вильямс, 2013 — 736 с.
19. Фаулер М. UML. Основы: пер. с англ./ Мартин Фаулер, Скотт Кендалл — СПб: Символ-Плюс, 2002. — 192 с.

ДОДАТОК А

Приклад XML-пакету тесту

```
<?xml version="1.0" encoding="utf-8"?>
<Test Name="Тест &amp;quot;Основи програмування&amp;quot;">
  <Question Text="Оберіть правильні ідентифікатори" Points="1" ChooseAllAnswers="1">
    <Answer IsRight="1">змінна</Answer>
    <Answer IsRight="1">@bool</Answer>
    <Answer IsRight="0">Number One</Answer>
    <Answer IsRight="1">@char</Answer>
    <Answer IsRight="1">_my_var</Answer>
  </Question>
  <Question Text="Якого значення набуде змінна?&amp;#xD;&amp;#xA;bool x = new bool();" Points="1">
    <Answer IsRight="0">null</Answer>
    <Answer IsRight="0">0</Answer>
    <Answer IsRight="1">false</Answer>
    <Answer IsRight="0">true</Answer>
  </Question>
  <Question Text="Оберіть бінарні операції" Points="2" ChooseAllAnswers="1">
    <Answer IsRight="0">логічне заперечення</Answer>
    <Answer IsRight="0">логічна операція</Answer>
    <Answer IsRight="1">остача від ділення</Answer>
    <Answer IsRight="0">декремент</Answer>
    <Answer IsRight="1">операція відношення</Answer>
  </Question>
  <Question Text="Оберіть правильну форму оголошення символьної константи" Points="1">
    <Answer IsRight="0">const int x = new int();</Answer>
    <Answer IsRight="1">const int I = 1;</Answer>
    <Answer IsRight="0">const int I;</Answer>
```

</Question>
</Test>

ДОДАТОК Б

Приклад збереженої процедури

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `tests_to_take`(IN idUser INT)
BEGIN
  DECLARE parent INT;
  DECLARE parentExists int DEFAULT FALSE;
  DECLARE done INT DEFAULT FALSE;
  DECLARE cur1 CURSOR FOR SELECT idParent FROM idUnits;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
  CREATE TEMPORARY TABLE idUnits (id INT, idParent INT);
  INSERT INTO idUnits (
        SELECT id, idParent FROM units WHERE id in (
              SELECT idObjectMain FROM link_objects AS linkUnitEntity
                    WHERE linkUnitEntity.IdObjectSub =
                          (SELECT idEntity FROM users WHERE id = idUser) ) );
  OPEN cur1;
  read_loop: LOOP
    SET parentExists = FALSE;
    FETCH cur1 INTO parent;
    IF done THEN
      LEAVE read_loop;
    END IF;
    WHILE (parent IS NOT NULL AND parentExists <> TRUE) DO
      INSERT INTO idUnits (SELECT id, idParent FROM units WHERE units.id=parent);
      SET parent = (SELECT idParent FROM units WHERE id=parent);
      SET parentExists = (SELECT COUNT(*) FROM idUnits WHERE id=parent);
    END WHILE;
  END LOOP;
  CLOSE cur1;
  CREATE TEMPORARY TABLE tests (id INT, idParent INT)
  SELECT test.* FROM test
    left join ap_datetime as startDate on (startDate.IdObject=test.Id)
```

```
    left join ap_datetime as endDate on (endDate.IdObject=test.Id)
    WHERE test.id in
        (SELECT link.IdObjectMain FROM link_objects AS link
                WHERE link.IdLink=2 AND link.IdReestrMain=3 AND
                        link.IdReestrSub=6 AND link.IdObjectSub IN
                                (SELECT id FROM idUnits )
        ) AND
    ifnull(startDate.IdReestr=3, true) and
    ifnull(startDate.IdLink=5, true) and
    ifnull(startDate.value <= now(), true) AND
    ifnull(endDate.IdReestr=3, true) and
    ifnull(endDate.IdLink=6, true) and
    ifnull(endDate.value >= now(), true);
  SET SQL_SAFE_UPDATES=0;
        DELETE t FROM tests t WHERE id in (
                SELECT survey.idTest FROM test_survey as survey
                WHERE t.idKind = 6 AND survey.idTest = t.id AND survey.idEntity =
                        (SELECT idEntity FROM users WHERE id = idUser));
  SET SQL_SAFE_UPDATES=1;
  SELECT * FROM tests;
  DROP temporary table if exists idUnits;
  DROP temporary table if exists test;
END
```

# ДОДАТОК В

## Код програми

### В.1 МОДЕЛЬ ТЕСТУ

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education_Tests_System.models
{
   public class Test: DbObject
   {
      private string name = string.Empty;
      private TestKind idKind;
      private int idAuthor;
      private Entity author;
      private TimeSpan? duration;
      private DateTime? beginDate;
      private DateTime? endDate;

      public string Name { get => name; set { name = value; OnPropertyChanged(nameof(Name)); } }
      public TestKind IdKind  { get => idKind; set => idKind = value; }
      public int IdAuthor { get => idAuthor; set { idAuthor = value; OnPropertyChanged(nameof(IdAuthor)); } }
      public TimeSpan? Duration
      {
         get
         {
            if (duration != null)
               return duration;
            duration = GetApTime(Consts.lnkTestDuration);
            return duration;
         }
         set { duration = value; OnPropertyChanged(nameof(IdAuthor)); }
      }

      public DateTime? BeginDate
      {
         get
         {
```

```
        if (beginDate != null)
            return beginDate;
        beginDate = GetApDateTime(Consts.lnkTestBeginDate);
        return beginDate;
    }
    set { beginDate = value; OnPropertyChanged(nameof(BeginDate)); }
}
public DateTime? EndDate
{
    get
    {
        if (endDate != null)
            return endDate;
        endDate = GetApDateTime(Consts.lnkTestEndDate);
        return endDate;
    }
    set { endDate = value; OnPropertyChanged(nameof(EndDate)); }
}

public int PassCount => DbHelper.GetTestPassCount(Id);
public int UniquePassCount => DbHelper.GetTestPassCount(Id, true);

public Entity Author
{
    get
    {
        if (author != null)
            return author;
        if (idAuthor != 0)
        {
            author = FindById(typeof(Entity), idAuthor) as Entity;
            if (!author.Loaded)
                author = null;
        }
        return author;
    }
}

private DbObjectList<Question> questions;
public DbObjectList<Question> Questions
{
    get
    {
        if (questions?.Loaded ?? false)
            return questions;
```

```
            questions = GetList(typeof(Question), "idTest", Id).ConvertAll(x => (Question)x);
            return questions;
        }
    }
    private DbObjectList<Unit> units;
    public DbObjectList<Unit> Units
    {
        get
        {
            if (units?.Loaded ?? false)
                return units;
            units = GetLinkedList(typeof(Unit), Consts.lnkTestUnit, true).ConvertAll(x => (Unit)x);
            return units;
        }
    }

    public override Dictionary<string, Type> GetPropsInfo()
    {
        if (propsInfo == null)
        {
            propsInfo = new Dictionary<string, Type>(base.GetPropsInfo());
            propsInfo.Add(nameof(Name), Name.GetType());
            propsInfo.Add(nameof(IdKind), IdKind.GetType());
            propsInfo.Add(nameof(IdAuthor), IdAuthor.GetType());
        }
        return propsInfo;
    }
    public override string TableName => Consts.tnTest;

    public override void CopyFrom(DbObject copy)
    {
        base.CopyFrom(copy);
        Name = (copy as Test).Name;
        IdKind = (copy as Test).IdKind;
        IdAuthor = (copy as Test).IdAuthor;
        author = null;
        units = null;
        questions = null;
    }
  }
}
```

## В.2 МОДЕЛЬ ПИТАННЯ

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education_Tests_System.models
{
   public class Question: DbObject
   {
      private string text = string.Empty;
      private int idTest;
      private float points=1;
      private bool? chooseAllAnswers;
      private bool isUserFinished;

      public string Text { get => text; set { text = value; OnPropertyChanged(nameof(Text));
} }
      public int IdTest { get => idTest; set => idTest = value; }
      public float Points { get => points; set { points = value;
OnPropertyChanged(nameof(Points)); } }
      public bool? ChooseAllAnswers
      {
         get
         {
            if (chooseAllAnswers != null)
               return chooseAllAnswers ?? false;
            chooseAllAnswers = GetApBool(Consts.lnkQuestionWithAllAnswersOpt);
            return chooseAllAnswers;
         }
         set { chooseAllAnswers = value;
OnPropertyChanged(nameof(ChooseAllAnswers)); }
      }
      public bool IsUserFinished { get => isUserFinished; set { isUserFinished = value;
OnPropertyChanged(nameof(IsUserFinished)); } }
      public int RightAnswersCount
      {
         get
         {
            int count = 0;
            foreach (var answer in Answers)
               if (answer.IsRight)
                  count++;
            return count;
```

```
        }
    }

    private DbObjectList<Answer> answers;
    public DbObjectList<Answer> Answers
    {
        get
        {
            if (answers?.Loaded ?? false)
                return answers;
            answers = GetList(typeof(Answer), "idQuestion", Id).ConvertAll(x =>
(Answer)x);
            return answers;
        }
    }

    public override Dictionary<string, Type> GetPropsInfo()
    {
        if (propsInfo == null)
        {
            propsInfo = new Dictionary<string, Type>(base.GetPropsInfo());
            propsInfo.Add(nameof(Text), Text.GetType());
            propsInfo.Add(nameof(IdTest), IdTest.GetType());
            propsInfo.Add(nameof(Points), Points.GetType());
        }
        return propsInfo;
    }
    public override string TableName => Consts.tnQuestion;

    public override void ResetAllLinks()
    {
        base.ResetAllLinks();
        answers = null;
    }
  }
}
```

### В.3 МОДЕЛЬ ВІДПОВІДІ

```
using System;
using System.Collections.Generic;
using System.Text;
```

```
namespace Education_Tests_System.models
{
   public class Answer: DbObject
   {
      private string text = string.Empty;
      private int idQuestion;
      private bool isRight;
      private bool isUserSelected;

      public string Text { get => text; set { text = value; OnPropertyChanged(nameof(Text));
} }
      public int IdQuestion { get => idQuestion; set => idQuestion = value; }
      public bool IsRight { get => isRight; set { isRight = value;
OnPropertyChanged(nameof(IsRight)); } }
      public bool IsUserSelected { get => isUserSelected; set { isUserSelected = value;
OnPropertyChanged(nameof(IsUserSelected)); } }

      public override Dictionary<string, Type> GetPropsInfo()
      {
         if (propsInfo == null)
         {
            propsInfo = new Dictionary<string, Type>(base.GetPropsInfo());
            propsInfo.Add(nameof(Text), Text.GetType());
            propsInfo.Add(nameof(IdQuestion), IdQuestion.GetType());
            propsInfo.Add(nameof(IsRight), IsRight.GetType());
         }
         return propsInfo;
      }
      public override string TableName => Consts.tnAnswer;
   }
}
```

## В.4 XAML РЕДАКТОРУ ТЕСТУ

```
<Window x:Class="Education_Tests_System.views.TestEditor"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:tool="http://schemas.xceed.com/wpf/xaml/toolkit"
      xmlns:local="clr-namespace:Education_Tests_System.views"
      xmlns:validators="clr-namespace:Education_Tests_System.validators"
      xmlns:System="clr-namespace:System;assembly=mscorlib"
```

```
    mc:Ignorable="d"
    WindowState="Maximized"
    Title="Редактор тесту" Height="500" Width="600"
    MinHeight="260" MinWidth="460"
    Closing="Window_Closing"
    Closed="Window_Closed">
  <Window.Resources>
    <Style x:Key="GridLabel" TargetType="Label">
      <Setter Property="VerticalAlignment" Value="Center"/>
      <Setter Property="Margin" Value="20, 10"/>
    </Style>
    <Style x:Key="GridTextBox" TargetType="TextBox" BasedOn="{StaticResource
BlackTextBox}">
      <Setter Property="HorizontalAlignment" Value="Left"/>
      <Setter Property="VerticalAlignment" Value="Center"/>
    </Style>
    <Style x:Key="GridComboBox" TargetType="ComboBox">
      <Setter Property="HorizontalAlignment" Value="Left"/>
      <Setter Property="VerticalAlignment" Value="Center"/>
    </Style>
    <Style x:Key="GridButton" TargetType="Button" BasedOn="{StaticResource
GreyButton}">
      <Setter Property="HorizontalAlignment" Value="Center"/>
      <Setter Property="VerticalAlignment" Value="Center"/>
      <Setter Property="Margin" Value="0, 10"/>
    </Style>
    <Style x:Key="GridCheckBox" TargetType="CheckBox" BasedOn="{StaticResource
BlackCheckBox}">
      <Setter Property="VerticalAlignment" Value="Center"/>
      <Setter Property="Margin" Value="20"/>
    </Style>
    <Style x:Key="ValidatedTextBox" TargetType="TextBox"
BasedOn="{StaticResource GridTextBox}">
      <Setter Property="Validation.ErrorTemplate">
        <Setter.Value>
          <ControlTemplate>
            <DockPanel LastChildFill="True">
              <Border BorderBrush="Red" BorderThickness="1">
                <AdornedElementPlaceholder/>
              </Border>
            </DockPanel>
          </ControlTemplate>
        </Setter.Value>
      </Setter>
      <Style.Triggers>
```

```
            <Trigger Property="Validation.HasError" Value="True">
              <Setter Property="ToolTip"
                Value="{Binding RelativeSource={RelativeSource Self},
                Path=(Validation.Errors)/ErrorContent}"/>
            </Trigger>
          </Style.Triggers>
        </Style>
        <DataTemplate x:Key="QuestionTab">
          <Grid>
            <Grid.ColumnDefinitions>
              <ColumnDefinition MaxWidth="170" Width="1*" />
              <ColumnDefinition MaxWidth="170" Width="1*" />
              <ColumnDefinition MaxWidth="170" Width="1*" />
              <ColumnDefinition Width="2*" />
            </Grid.ColumnDefinitions>
            <Grid.RowDefinitions>
              <RowDefinition Height="70" />
              <RowDefinition Height="60"  />
              <RowDefinition />
              <RowDefinition Height="50" />
            </Grid.RowDefinitions>
            <Label Grid.Column="0" Grid.Row="0" Content="Питання"
Style="{StaticResource GridLabel}" />
            <Label Grid.Column="0" Grid.Row="1" Content="Бали"
Style="{StaticResource GridLabel}" />
            <TextBox Grid.Column="1" Grid.Row="0" Grid.ColumnSpan="3"
Style="{StaticResource ValidatedTextBox}" TextWrapping="Wrap"
AcceptsReturn="True"
                MinWidth="220" MinHeight="60" Name="txtText" Margin=" 0 5 10 0"
HorizontalAlignment="Stretch">
              <TextBox.Text>
                <Binding Path="Text" Mode="TwoWay"
UpdateSourceTrigger="PropertyChanged">
                  <Binding.ValidationRules>
                    <validators:NotEmptyValidator ValidatesOnTargetUpdated="True"/>
                  </Binding.ValidationRules>
                </Binding>
              </TextBox.Text>
            </TextBox>
            <tool:DoubleUpDown Grid.Column="1" Grid.Row="1" Name="PointSpan"
HorizontalAlignment="Left" VerticalAlignment="Center"
                  Width="50" MinWidth="40" MaxWidth="80" Minimum="1"
Value="{Binding Path=Points, Mode=TwoWay,
UpdateSourceTrigger=PropertyChanged}"/>
```

```
        <CheckBox Grid.Column="3" Grid.Row="1" HorizontalAlignment="Right"
Style="{StaticResource GridCheckBox}"
                  Name="AllAnswerOpt" IsChecked="{Binding Path=ChooseAllAnswers,
Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}">оберіть усі
відповіді</CheckBox>
        <GroupBox Grid.Column="0" Grid.Row="2" Grid.ColumnSpan="4"
                  Header="Відповіді" Padding="0 5">
          <Grid>
            <DataGrid CellStyle="{StaticResource NonSelectionBgCell}"
RowHeight="40" AutoGenerateColumns="False" CanUserAddRows="False"
SelectionUnit="Cell"
                      Name="AnswerGrid" MinHeight="100" ItemsSource="{Binding
Path=Answers,                                                    Mode=OneWay}"
CurrentCellChanged="AnswerGrid_CurrentCellChanged">
              <DataGrid.Columns>
                <DataGridTemplateColumn Header="Вірна">
                  <DataGridTemplateColumn.CellTemplate>
                    <DataTemplate>
                      <CheckBox IsChecked="{Binding Path=IsRight,
Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}" />
                    </DataTemplate>
                  </DataGridTemplateColumn.CellTemplate>
                </DataGridTemplateColumn>
                <DataGridTemplateColumn Header="Текст відповіді" Width="9*">
                  <DataGridTemplateColumn.CellTemplate>
                    <DataTemplate>
                      <TextBox TextWrapping="Wrap" AcceptsReturn="True"
HorizontalScrollBarVisibility="Disabled" VerticalScrollBarVisibility="Auto"
                               Text="{Binding Path=Text, Mode=TwoWay,
UpdateSourceTrigger=PropertyChanged}" />
                    </DataTemplate>
                  </DataGridTemplateColumn.CellTemplate>
                </DataGridTemplateColumn>
                <DataGridTemplateColumn>
                  <DataGridTemplateColumn.CellTemplate>
                    <DataTemplate>
                      <Button Style="{StaticResource DataGridButton}"
Tag="{Binding Path=Id}"
                              Name="DeleteAnswer" Click="DeleteAnswer_Click">
                        <Image Height="20" Width="20" Margin="2"
Source="{StaticResource Delete}" />
                      </Button>
                    </DataTemplate>
                  </DataGridTemplateColumn.CellTemplate>
                </DataGridTemplateColumn>
```

```
                </DataGrid.Columns>
              </DataGrid>
            </Grid>
          </GroupBox>
          <StackPanel Grid.Column="0" Grid.Row="3" Grid.ColumnSpan="4"
Orientation="Horizontal" HorizontalAlignment="Center">
            <Button Click="MoveToQuestion_Click" Margin="10 0">
              <Button.Style>
                <Style TargetType="Button" BasedOn="{StaticResource GridButton}">
                  <Setter Property="IsEnabled" Value="True"/>
                  <Setter Property="Opacity" Value="1"/>
                  <Style.Triggers>
                    <DataTrigger Binding="{Binding Path=(Validation.HasError),
ElementName=txtText}" Value="True">
                      <Setter Property="IsEnabled" Value="False"/>
                      <Setter Property="Opacity" Value="0.5"/>
                    </DataTrigger>
                    <DataTrigger Binding="{Binding Items.Count,
ElementName=AnswerGrid}" Value="0">
                      <Setter Property="IsEnabled" Value="False"/>
                      <Setter Property="Opacity" Value="0.5"/>
                    </DataTrigger>
                  </Style.Triggers>
                </Style>
              </Button.Style>
              <StackPanel Orientation="Horizontal">
                <Image Source="{StaticResource Next}" Width="20" Height="20"
Margin="0, 0, 5, 0" />
                <TextBlock Text="Наступне питання" VerticalAlignment="Center" />
              </StackPanel>
            </Button>
            <Button Margin="10 0" Click="Save_Click">
              <Button.Style>
                <Style TargetType="Button" BasedOn="{StaticResource GridButton}">
                  <Setter Property="IsEnabled" Value="True"/>
                  <Setter Property="Opacity" Value="1"/>
                  <Style.Triggers>
                    <DataTrigger Binding="{Binding Path=(Validation.HasError),
ElementName=txtText}" Value="True">
                      <Setter Property="IsEnabled" Value="False"/>
                      <Setter Property="Opacity" Value="0.5"/>
                    </DataTrigger>
                    <DataTrigger Binding="{Binding Items.Count,
ElementName=AnswerGrid}" Value="0">
                      <Setter Property="IsEnabled" Value="False"/>
```

```
                        <Setter Property="Opacity" Value="0.5"/>
                    </DataTrigger>
                </Style.Triggers>
            </Style>
        </Button.Style>
        <StackPanel Orientation="Horizontal">
            <Image Source="{StaticResource Save}" Width="20" Height="20"
Margin="0, 0, 5, 0" />
            <TextBlock Text="Зберегти" VerticalAlignment="Center" />
        </StackPanel>
    </Button>
</StackPanel>
</Grid>
</DataTemplate>
</Window.Resources>
<TabControl Name="TestTab" TabStripPlacement="Left"
SelectionChanged="TestTab_SelectionChanged">
<TabItem Header="Загальне" Name="GeneralTab">
<TabItem.Content>
<Grid MinHeight="470" MinWidth="350">
<Grid.ColumnDefinitions>
<ColumnDefinition MaxWidth="200" Width="1*" />
<ColumnDefinition Width="2*" />
</Grid.ColumnDefinitions>
<Grid.RowDefinitions>
<RowDefinition Height="50" />
<RowDefinition Height="50" />
<RowDefinition Height="60" />
<RowDefinition Height="60" />
<RowDefinition />
<RowDefinition Height="50" />
</Grid.RowDefinitions>
<Label Grid.Column="0" Grid.Row="0" Content="Назва"
Style="{StaticResource GridLabel}" />
<Label Grid.Column="0" Grid.Row="1" Content="Тип"
Style="{StaticResource GridLabel}" />
<TextBox Grid.Column="1" Grid.Row="0" Style="{StaticResource
ValidatedTextBox}"
MinWidth="220" MinHeight="40" Name="txtName" Margin=" 0 5 10 0"
HorizontalAlignment="Stretch">
<TextBox.Text>
<Binding Path="Name" Mode="TwoWay"
UpdateSourceTrigger="PropertyChanged">
<Binding.ValidationRules>
```

```
                        <validators:NotEmptyValidator
ValidatesOnTargetUpdated="True"/>
                    </Binding.ValidationRules>
                </Binding>
            </TextBox.Text>
        </TextBox>
        <ComboBox Grid.Column="1" Grid.Row="1" Style="{StaticResource
GridComboBox}"
                  MinWidth="110" Name="cmbKind" AllowDrop="False"
SelectionChanged="CmbKind_SelectionChanged">
            <ComboBoxItem Name="Trainy">Тренувальний</ComboBoxItem>
            <ComboBoxItem Name="Exam">Контрольний</ComboBoxItem>
        </ComboBox>
        <CheckBox Grid.Column="0" Grid.Row="2" Style="{StaticResource
GridCheckBox}"
                  Name="PeriodRestrict">обмеження періоду</CheckBox>
        <CheckBox Grid.Column="0" Grid.Row="3" Style="{StaticResource
GridCheckBox}"
                  Name="TimeRestrict">обмеження часу</CheckBox>
        <StackPanel Grid.Column="1" Grid.Row="2" Orientation="Horizontal"
VerticalAlignment="Center" HorizontalAlignment="Left"
                    MinWidth="210" IsEnabled="{Binding ElementName=PeriodRestrict,
Path=IsChecked}">
            <DatePicker SelectedDateFormat="Short" Name="BeginDate"
                        SelectedDate="{Binding Path=BeginDate, Mode=TwoWay,
UpdateSourceTrigger=PropertyChanged, TargetNullValue={x:Static
System:DateTime.Now}}"/>
            <Label>-</Label>
            <DatePicker SelectedDateFormat="Short" Name="EndDate"
                        SelectedDate="{Binding Path=EndDate, Mode=TwoWay,
UpdateSourceTrigger=PropertyChanged, TargetNullValue={x:Static
System:DateTime.MaxValue}}"/>
        </StackPanel>
        <tool:TimeSpanUpDown Name="TimeSpan" Grid.Column="1"
Grid.Row="3" HorizontalAlignment="Left" VerticalAlignment="Center"
                             Width="75" MinWidth="60" MaxWidth="100"
Minimum="0:5:00" DefaultValue="0:25:00" Watermark="0:25:00"
                             Value="{Binding Path=Duration, Mode=TwoWay,
UpdateSourceTrigger=PropertyChanged}" IsEnabled="{Binding
ElementName=TimeRestrict, Path=IsChecked}" />
        <GroupBox Grid.Column="0" Grid.Row="4" Grid.ColumnSpan="2"
                  Header="Доступний для" Padding="0 5">
            <Grid>
                <Grid.RowDefinitions>
                    <RowDefinition />
```

```
                    <RowDefinition Height="50"/>
                </Grid.RowDefinitions>
                <DataGrid CellStyle="{StaticResource NonBorderCell}"
AutoGenerateColumns="False" CanUserAddRows="False"
                    Name="UnitGrid" MinHeight="100">
                    <DataGrid.Columns>
                        <DataGridTemplateColumn Header="Організаційна одиниця">
                            <DataGridTemplateColumn.CellTemplate>
                                <DataTemplate>
                                    <TextBlock Text="{Binding Path=DeepDescription}" />
                                </DataTemplate>
                            </DataGridTemplateColumn.CellTemplate>
                        </DataGridTemplateColumn>
                        <DataGridTemplateColumn>
                            <DataGridTemplateColumn.CellTemplate>
                                <DataTemplate>
                                    <Button Style="{StaticResource DataGridButton}"
Tag="{Binding Path=Id}"
                                        Name="DeleteUnit" Click="DeleteUnit_Click">
                                        <Image Height="20" Width="20" Margin="2"
Source="{StaticResource Delete}" />
                                    </Button>
                                </DataTemplate>
                            </DataGridTemplateColumn.CellTemplate>
                        </DataGridTemplateColumn>
                    </DataGrid.Columns>
                </DataGrid>
                <Button Grid.Row="1" MinHeight="30" Style="{StaticResource
GridButton}"
                    Name="AddUnit" Click="AddUnit_Click">
                    <StackPanel Orientation="Horizontal">
                        <Image Source="{StaticResource New}" Width="20" Height="20"
Margin="0, 0, 5, 0" />
                        <TextBlock Text="Додати" VerticalAlignment="Center" />
                    </StackPanel>
                </Button>
            </Grid>
        </GroupBox>
        <StackPanel Grid.Column="0" Grid.Row="5" Grid.ColumnSpan="2"
Orientation="Horizontal" HorizontalAlignment="Center">
            <Button Name="MoveToQuestion" Margin="10 0"
Click="MoveToQuestion_Click">
                <Button.Style>
                    <Style TargetType="Button" BasedOn="{StaticResource
GridButton}">
```

```
                        <Setter Property="IsEnabled" Value="True"/>
                        <Setter Property="Opacity" Value="1"/>
                        <Style.Triggers>
                          <DataTrigger Binding="{Binding Path=(Validation.HasError), ElementName=txtName}" Value="True">
                            <Setter Property="IsEnabled" Value="False"/>
                            <Setter Property="Opacity" Value="0.5"/>
                          </DataTrigger>
                          <DataTrigger Binding="{Binding Items.Count, ElementName=UnitGrid}" Value="0">
                            <Setter Property="IsEnabled" Value="False"/>
                            <Setter Property="Opacity" Value="0.5"/>
                          </DataTrigger>
                        </Style.Triggers>
                      </Style>
                    </Button.Style>
                    <StackPanel Orientation="Horizontal">
                      <Image Source="{StaticResource Next}" Width="20" Height="20" Margin="0, 0, 5, 0" />
                      <TextBlock Text="Перейти до питань" VerticalAlignment="Center" />
                    </StackPanel>
                  </Button>
                  <Button Name="SaveGeneral" Margin="10 0" Click="Save_Click">
                    <Button.Style>
                      <Style TargetType="Button" BasedOn="{StaticResource GridButton}">
                        <Setter Property="IsEnabled" Value="True"/>
                        <Setter Property="Opacity" Value="1"/>
                        <Style.Triggers>
                          <DataTrigger Binding="{Binding Path=(Validation.HasError), ElementName=txtName}" Value="True">
                            <Setter Property="IsEnabled" Value="False"/>
                            <Setter Property="Opacity" Value="0.5"/>
                          </DataTrigger>
                          <DataTrigger Binding="{Binding Items.Count, ElementName=UnitGrid}" Value="0">
                            <Setter Property="IsEnabled" Value="False"/>
                            <Setter Property="Opacity" Value="0.5"/>
                          </DataTrigger>
                        </Style.Triggers>
                      </Style>
                    </Button.Style>
                    <StackPanel Orientation="Horizontal">
```

```
                    <Image Source="{StaticResource Save}" Width="20" Height="20"
Margin="0, 0, 5, 0" />
                    <TextBlock Text="Зберегти" VerticalAlignment="Center" />
                  </StackPanel>
                </Button>
              </StackPanel>
            </Grid>
          </TabItem.Content>
        </TabItem>
    </TabControl>
</Window>
```

## В.5 С# РЕДАКТОРУ ТЕСТУ

```
using Education_Tests_System.models;
using System;
using System.Linq;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;

namespace Education_Tests_System.views
{
    /// <summary>
    /// Логика взаимодействия для TestEditor.xaml
    /// </summary>
    public partial class TestEditor : Window
    {
        private User user;
        private Test test;
        private Test testCopy;

        public TestEditor(User user)
        {
            InitializeComponent();

            this.user = user;
            test = new Test();
            testCopy = new Test();
            testCopy.Units.Loaded = true;
            testCopy.IdKind = TestKind.Trainy;
            testCopy.IdAuthor = user.IdEntity;
            GeneralTab.DataContext = testCopy;
```

```
        UnitGrid.ItemsSource = testCopy.Units;

        PrepareControls();
    }

    public TestEditor(Test test)
    {
        InitializeComponent();

        this.test = test;
        testCopy = new Test();
        testCopy.CopyFrom(this.test);
        GeneralTab.DataContext = testCopy;
        UnitGrid.ItemsSource = testCopy.Units;

        PrepareControls();
    }

    private void PrepareControls()
    {
        BeginDate.DisplayDateStart = DateTime.Today;
        EndDate.DisplayDateStart = DateTime.Today;
        LoadArtificialBinding();
    }
    private void LoadArtificialBinding(bool reload = false)
    {
        PeriodRestrict.IsChecked = testCopy.BeginDate != null || testCopy.EndDate != null;
        TimeRestrict.IsChecked = testCopy.Duration != null;
        CmbKind_SetSelectedItem(testCopy);

        if (reload)
            for (int i = TestTab.Items.Count - 1; i > 0; i--)
            {
                TabItem tab = TestTab.Items[i] as TabItem;
                if (tab.ContentTemplate == (DataTemplate)FindResource("QuestionTab"))
                    TestTab.Items.Remove(tab);
            }
        foreach (var question in testCopy.Questions)
            AddNewQuestionTab(question);
    }

    private void Window_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
    { e.Cancel = !AffirmBeforeClose(); }
```

```
        private bool AffirmBeforeClose()
        {
            MessageBoxResult result = MessageBox.Show("Закрити редактор? Усі незбережені зміни не застосуються", "Підтвердіть дію", MessageBoxButton.YesNo, MessageBoxImage.Question);
            return result == MessageBoxResult.Yes;
        }

        private void Window_Closed(object sender, EventArgs e) => Owner?.Show();

        private void CmbKind_SetSelectedItem(Test test)
        {
            if (test.IdKind == TestKind.Trainy)
                Trainy.IsSelected = true;
            else
                Exam.IsSelected = true;
        }
        private void CmbKind_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            ComboBoxItem item = (sender as ComboBox).SelectedItem as ComboBoxItem;
            testCopy.IdKind = item == Trainy ? TestKind.Trainy : TestKind.Exam;
            TimeRestrict.IsEnabled = item == Exam;
        }

        private void AddUnit_Click(object sender, RoutedEventArgs e)
        {
            // mock  adding 'IT-51' to list
            if (!testCopy.Units.Any(item => item.Id == 89))
                testCopy.Units.Add(DbObject.FindById(typeof(Unit), 89) as Unit);
        }

        private void DeleteUnit_Click(object sender, RoutedEventArgs e)
        {
            int id = Convert.ToInt32((sender as Control).Tag);
            testCopy.Units.Remove(testCopy.Units.First(item => item.Id == id));
        }

        private void DeleteAnswer_Click(object sender, RoutedEventArgs e)
        {
            Question question = (TestTab.SelectedItem as TabItem).DataContext as Question;
            int id = Convert.ToInt32((sender as Control).Tag);
            question.Answers.Remove(question.Answers.First(item => item.Id == id));
        }
```

```
private void MoveToQuestion_Click(object sender, RoutedEventArgs e)
{
   int tabIndex = Convert.ToInt32(TestTab.SelectedIndex);
   if (tabIndex == TestTab.Items.Count - 1)
   {
      Question question = new Question();
      testCopy.Questions.Loaded = true;
      testCopy.Questions.Add(question);
      AddNewQuestionTab(question);
   }
   TestTab.SelectedIndex = tabIndex + 1;
}
private void AddNewQuestionTab(Question question)
{
   TabItem questionTab = new TabItem
   {
      Header = "Питання №" + TestTab.Items.Count,
      ContentTemplate = (DataTemplate)FindResource("QuestionTab"),
      DataContext = question,
   };
   Binding binding = new Binding();
   binding.RelativeSource = RelativeSource.Self;
   binding.Path = new PropertyPath("DataContext");
   questionTab.SetBinding(ContentProperty, binding);
   TestTab.Items.Add(questionTab);
}

private void TestTab_SelectionChanged(object sender, SelectionChangedEventArgs
e)
{
   TabItem tab = (sender as TabControl).SelectedItem as TabItem;
   if (tab != null && tab.DataContext != null && tab.DataContext is Question)
   {
      Question question = tab.DataContext as Question;
      if (question.Answers.Count == 0 || question.Answers.Last().Text != string.Empty)
      {
         question.Answers.Loaded = true;
         question.Answers.Add(new Answer());
      }
   }
}

private void AnswerGrid_CurrentCellChanged(object sender, EventArgs e)
{
   DataGrid grid = sender as DataGrid;
```

```
        if (grid.DataContext != null && grid.DataContext is Question)
        {
            Question question = grid.DataContext as Question;
            if (question.Answers.Count == 0 || question.Answers.Last().Text != string.Empty)
                question.Answers.Add(new Answer());
        }
    }

    private void Save_Click(object sender, RoutedEventArgs e)
    {
        try
        {
            DbHelper.BeginTransaction();
            if (user != null && !user.Entity.Tests.Contains(test))
                user.Entity.Tests.Add(test);
            test.CopyFrom(testCopy);
            test.Save();
            if (testCopy.Id <= 0)
                testCopy.Id = test.Id;

            SynchUnits();
            SynchQuestions();

            // resolve ap
            if (PeriodRestrict.IsChecked ?? false)
            {
                bool isNew = test.BeginDate == null;
                if (testCopy.BeginDate == null)
                    testCopy.BeginDate = BeginDate.SelectedDate;
                test.BeginDate = testCopy.BeginDate;
                test.UpdateAp(Consts.tnApDateTime,                Consts.lnkTestBeginDate,
test.BeginDate, isNew);

                isNew = test.EndDate == null;
                if (testCopy.EndDate == null)
                    testCopy.EndDate = EndDate.SelectedDate;
                test.EndDate = testCopy.EndDate;
                test.UpdateAp(Consts.tnApDateTime,  Consts.lnkTestEndDate,  test.EndDate,
isNew);
            }
            else if (PeriodRestrict.IsChecked != null)
            {
                if (test.BeginDate != null)
                    test.DeleteAp(Consts.tnApDateTime, Consts.lnkTestBeginDate);
                if (test.EndDate != null)
```

```
                test.DeleteAp(Consts.tnApDateTime, Consts.lnkTestEndDate);
            }

            if (TimeRestrict.IsChecked ?? false)
            {
                bool isNew = test.Duration == null;
                if (testCopy.Duration == null)
                    testCopy.Duration = TimeSpan.DefaultValue;
                test.Duration = testCopy.Duration;
                test.UpdateAp(Consts.tnApTime, Consts.lnkTestDuration, test.Duration,
isNew);
            }
            else if (TimeRestrict.IsChecked != null && test.Duration != null)
                test.DeleteAp(Consts.tnApTime, Consts.lnkTestDuration);
            DbHelper.CommitTransaction();
        }
        catch
        {
            DbHelper.RollbackTransaction();
            if (test.Id > 0)
            {
                test.LoadByProp("id", test.Id);
                test.ResetAllLinks();
                testCopy.CopyFrom(test);
                UnitGrid.ItemsSource = testCopy.Units;
            }
            else
            {
                test = new Test();
                testCopy.CopyFrom(test);
                testCopy.Units.Loaded = true;
                UnitGrid.ItemsSource = testCopy.Units;
            }
            LoadArtificialBinding(true);
            MessageBox.Show("Вибачте, під час збереження даних до бази виникла
помилка", "Увага", MessageBoxButton.OK, MessageBoxImage.Error);
        }
    }

    private void SynchUnits()
    {
        for (int i = test.Units.Count - 1; i >= 0; i--)
        {
            Unit unit = test.Units[i];
            if (!testCopy.Units.Any(item => item.Id == unit.Id))
```

```
            {
                test.DeleteLink(Consts.tnUnit, unit.Id, Consts.lnkTestUnit, true);
                test.Units.Remove(unit);
            }
        }
        foreach (var unit in testCopy.Units)
            if (!test.Units.Any(item => item.Id == unit.Id))
            {
                test.AddLink(Consts.tnUnit, unit.Id, Consts.lnkTestUnit, true);
                test.Units.Add(unit);
            }
    }
    private void SynchQuestions()
    {
        for (int i = testCopy.Questions.Count - 1; i >= 0; i--)
        {
            Question question = testCopy.Questions[i];
            if (question.Answers.Where(item => item.Text != string.Empty).Count() == 0)
            {
                if (question.Id > 0)
                    question.Delete();
                testCopy.Questions.Remove(question);
            }
        }
        for (int i = test.Questions.Count-1; i >= 0; i--)
        {
            Question question = test.Questions[i];
            if (!testCopy.Questions.Any(item => item.Id == question.Id))
            {
                test.Questions.Remove(question);
                question.Delete();
                question.DeleteAp(Consts.tnApBool,
Consts.lnkQuestionWithAllAnswersOpt);
                foreach (var answer in question.Answers)
                    answer.Delete();
            }
        }
        foreach (var question in testCopy.Questions)
        {
            question.IdTest = test.Id;
            question.Save();
            Question baseQuestion = null;
            if (test.Questions.Any(item => item.Id == question.Id))
                baseQuestion = test.Questions.First(item => item.Id == question.Id);
            else
```

```
                test.Questions.Add(question);
            if (question.ChooseAllAnswers ?? false)
            {
                bool isNew = baseQuestion == null || baseQuestion.ChooseAllAnswers == null;
                if (baseQuestion != null)
                    baseQuestion.ChooseAllAnswers = question.ChooseAllAnswers;
                question.UpdateAp(Consts.tnApBool,
Consts.lnkQuestionWithAllAnswersOpt, question.ChooseAllAnswers, isNew);
            }
            else if (question.ChooseAllAnswers != null)
                question.DeleteAp(Consts.tnApBool,
Consts.lnkQuestionWithAllAnswersOpt);

            if (baseQuestion != null)
                baseQuestion.Answers.Where(answer => !question.Answers.Any(item =>
item.Id == answer.Id)).ToList()
                                .ForEach(answer => answer.Delete());

            for (int i = question.Answers.Count - 1; i >= 0; i--)
            {
                Answer answer = question.Answers[i];
                answer.IdQuestion = question.Id;
                if (answer.Text == string.Empty)
                    question.Answers.Remove(answer);
                else
                    answer.Save();
            }
        }
    }
}
```

## В.6 XAML ПРОХОДЖЕННЯ ТЕСТУ

```
<Window x:Class="Education_Tests_System.views.TakeSurvey"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Education_Tests_System.views"
        mc:Ignorable="d"
        Closed="Window_Closed"
        Title="Проходження тесту"
```

```
    WindowStartupLocation="CenterScreen"
    ResizeMode="NoResize" Height="480" Width="640">
  <Window.Resources>
    <Style x:Key="GridButton" TargetType="Button" BasedOn="{StaticResource GreyButton}">
      <Setter Property="HorizontalAlignment" Value="Center"/>
      <Setter Property="VerticalAlignment" Value="Center"/>
      <Setter Property="Margin" Value="0, 10"/>
    </Style>
    <Style x:Key="NonBorderTextBox" TargetType="TextBox">
      <Setter Property="BorderBrush" Value="{x:Null}" />
      <Setter Property="Template">
        <Setter.Value>
          <ControlTemplate TargetType="{x:Type TextBox}">
            <Border x:Name="border" BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="{TemplateBinding BorderThickness}" Background="{TemplateBinding Background}" SnapsToDevicePixels="True">
              <ScrollViewer x:Name="PART_ContentHost" Focusable="false" HorizontalScrollBarVisibility="Hidden" VerticalScrollBarVisibility="Hidden"/>
            </Border>
            <ControlTemplate.Triggers>
              <Trigger Property="IsMouseOver" Value="True">
                <Setter Property="BorderBrush" Value="{x:Null}" />
              </Trigger>
              <Trigger Property="IsFocused" Value="True">
                <Setter Property="BorderBrush" Value="{x:Null}" />
              </Trigger>
              <Trigger Property="IsEnabled" Value="False">
                <Setter Property="BorderBrush" Value="{x:Null}" />
              </Trigger>
            </ControlTemplate.Triggers>
          </ControlTemplate>
        </Setter.Value>
      </Setter>
    </Style>
    <DataTemplate x:Key="QuestionTab">
      <Grid Margin="20 10 20 0">
        <Grid.RowDefinitions>
          <RowDefinition Height="Auto"/>
          <RowDefinition />
        </Grid.RowDefinitions>
        <StackPanel Grid.Row="0" Orientation="Vertical">
          <TextBox IsReadOnly="True" Focusable="False" Cursor="Arrow" Style="{StaticResource NonBorderTextBox}" FontWeight="Bold"
```

```
                Text="{Binding Path=Text, Mode=OneWay, UpdateSourceTrigger=PropertyChanged}" />
            <DockPanel Margin="0 5">
              <StackPanel Orientation="Horizontal" DockPanel.Dock="Left">
                <Label Content="Оберіть"/>
                <Label Name="AnswerCount">
                  <Label.Style>
                    <Style TargetType="Label">
                      <Setter Property="Content" Value="{Binding Path=RightAnswersCount, Mode=OneWay, UpdateSourceTrigger=PropertyChanged}" />
                      <Style.Triggers>
                        <DataTrigger Binding="{Binding Path=ChooseAllAnswers}" Value="True">
                          <Setter Property="Content" Value="yci" />
                        </DataTrigger>
                      </Style.Triggers>
                    </Style>
                  </Label.Style>
                </Label>
              </StackPanel>
              <StackPanel Orientation="Horizontal" DockPanel.Dock="Right" HorizontalAlignment="Right">
                <Label Content="Бали"/>
                <Label Name="Points" Content="{Binding Path=Points, Mode=OneWay, UpdateSourceTrigger=PropertyChanged}" />
              </StackPanel>
            </DockPanel>
          </StackPanel>
          <Grid Grid.Row="1" Margin="0 10">
            <DataGrid RowHeight="40" AutoGenerateColumns="False" CanUserAddRows="False" CellStyle="{StaticResource NonSelectionBgCell}"
                      SelectionUnit="Cell" GridLinesVisibility="None" Background="Transparent" HeadersVisibility="None" BorderBrush="Transparent"
                      Name="AnswerGrid" ItemsSource="{Binding Path=Answers, Mode=OneWay}">
              <DataGrid.Style>
                <Style TargetType="DataGrid">
                  <Style.Triggers>
                    <DataTrigger Binding="{Binding Path=IsUserFinished}" Value="True">
                      <Setter Property="IsEnabled" Value="False" />
                    </DataTrigger>
                  </Style.Triggers>
                </Style>
              </DataGrid.Style>
```

```
                <DataGrid.Columns>
                    <DataGridTemplateColumn>
                        <DataGridTemplateColumn.CellTemplate>
                            <DataTemplate>
                                <CheckBox Tag="{Binding Path=Id, Mode=OneWay}"
IsChecked="{Binding Path=IsUserSelected, Mode=TwoWay,
UpdateSourceTrigger=PropertyChanged}" Checked="Answer_Click"
Unchecked="Answer_Click" />
                            </DataTemplate>
                        </DataGridTemplateColumn.CellTemplate>
                    </DataGridTemplateColumn>
                    <DataGridTemplateColumn>
                        <DataGridTemplateColumn.CellTemplate>
                            <DataTemplate>
                                <TextBox IsReadOnly="True" Focusable="False"
Cursor="Arrow" Style="{StaticResource NonBorderTextBox}"
HorizontalScrollBarVisibility="Disabled" VerticalScrollBarVisibility="Auto"
                                    Text="{Binding Path=Text, Mode=OneWay,
UpdateSourceTrigger=PropertyChanged}" />
                            </DataTemplate>
                        </DataGridTemplateColumn.CellTemplate>
                    </DataGridTemplateColumn>
                    <DataGridTemplateColumn>
                        <DataGridTemplateColumn.CellTemplate>
                            <DataTemplate>
                                <Image Name="ImgResult" Height="20" Width="20"
Margin="10 0" VerticalAlignment="Top">
                                    <Image.Style>
                                        <Style TargetType="Image">
                                            <Setter Property="Visibility" Value="Hidden" />
                                            <Style.Triggers>
                                                <DataTrigger Binding="{Binding Path=IsRight}"
Value="True">
                                                    <Setter Property="Source" Value="{StaticResource
Right}" />
                                                </DataTrigger>
                                                <DataTrigger Binding="{Binding Path=IsRight}"
Value="False">
                                                    <Setter Property="Source" Value="{StaticResource
Wrong}" />
                                                </DataTrigger>
                                                <MultiDataTrigger>
                                                    <MultiDataTrigger.Conditions>
                                                        <Condition Binding="{Binding
ElementName=AnswerGrid, Path=IsEnabled}" Value="False" />
```

```
                                        <Condition Binding="{Binding
Path=IsUserSelected}" Value="True" />
                                    </MultiDataTrigger.Conditions>
                                    <Setter Property="Visibility" Value="Visible" />
                                </MultiDataTrigger>
                            </Style.Triggers>
                        </Style>
                    </Image.Style>
                </Image>
            </DataTemplate>
        </DataGridTemplateColumn.CellTemplate>
    </DataGridTemplateColumn>
</DataGrid.Columns>
</DataGrid>
</Grid>
</Grid>
</DataTemplate>
</Window.Resources>
<Grid>
<Grid.RowDefinitions>
<RowDefinition />
<RowDefinition Height="45"/>
<RowDefinition Height="30"/>
</Grid.RowDefinitions>
<TabControl Grid.Row="0" Name="TestSurveyTab" TabStripPlacement="Top"
BorderBrush="GhostWhite" SelectionChanged="TestSurveyTab_SelectionChanged">
<TabControl.Resources>
<Style TargetType="TabItem">
<Setter Property="Template">
<Setter.Value>
<ControlTemplate TargetType="TabItem">
<Grid Name="Panel" Margin="2 0">
<ContentPresenter x:Name="ContentSite"
VerticalAlignment="Center"
HorizontalAlignment="Center"
ContentSource="Header"
Margin="10" />
</Grid>
<ControlTemplate.Triggers>
<Trigger Property="IsSelected" Value="False">
<Setter TargetName="Panel" Property="Background"
Value="GhostWhite" />
</Trigger>
<Trigger Property="IsSelected" Value="True">
<Setter TargetName="Panel" Property="Background">
```

```
                        <Setter.Value>
                            <LinearGradientBrush EndPoint="0.5,1" MappingMode="RelativeToBoundingBox" StartPoint="0.5,0">
                                <GradientStop Color="White" Offset="1"/>
                                <GradientStop Color="#B2A2A2A2" Offset="0"/>
                            </LinearGradientBrush>
                        </Setter.Value>
                    </Setter>
                </Trigger>
            </ControlTemplate.Triggers>
        </ControlTemplate>
    </Setter.Value>
</Setter>
</Style>
</TabControl.Resources>
</TabControl>
<StackPanel Grid.Row="1" Orientation="Horizontal" HorizontalAlignment="Center">
    <Button Name="BtnCheck" Margin="10 0" Click="Check_Click" Content="Перевірити">
        <Button.Style>
            <Style TargetType="Button" BasedOn="{StaticResource GridButton}">
                <Setter Property="Visibility" Value="Visible" />
                <Style.Triggers>
                    <DataTrigger Binding="{Binding Path=IdKind, Mode=OneWay, UpdateSourceTrigger=PropertyChanged}" Value="6">
                        <Setter Property="Visibility" Value="Hidden" />
                    </DataTrigger>
                </Style.Triggers>
            </Style>
        </Button.Style>
    </Button>
    <Button Margin="10 0" Click="Previous_Click" Style="{StaticResource GridButton}">
        <StackPanel Orientation="Horizontal">
            <Image Source="{StaticResource Next}" FlowDirection="RightToLeft" Width="20" Height="20" Margin="0, 0, 8, 0" />
            <TextBlock Text="Назад" VerticalAlignment="Center" />
        </StackPanel>
    </Button>
    <Button Margin="10 0" Click="Next_Click" Style="{StaticResource GridButton}">
        <StackPanel Orientation="Horizontal">
            <TextBlock Text="Далі" VerticalAlignment="Center" />
```

```
                <Image Source="{StaticResource Next}" Width="20" Height="20"
Margin="8, 0, 0, 0" />
            </StackPanel>
        </Button>
    </StackPanel>
    <DockPanel Grid.Row="2" >
        <StackPanel Orientation="Horizontal" VerticalAlignment="Bottom"
HorizontalAlignment="Left" Margin="10 0"
                    Name="TimePanel">
            <Label Content="Залишилось часу:"/>
            <Label Name="TimeLeft"/>
        </StackPanel>
        <StackPanel Orientation="Horizontal" VerticalAlignment="Bottom"
HorizontalAlignment="Right" Margin="10 0">
            <Label Content="Пройдено:"/>
            <Label Name="QuestionReadyCount" Content="{Binding TakenQuestionCount,
Mode=OneWay, UpdateSourceTrigger=PropertyChanged}"/>
        </StackPanel>
    </DockPanel>
  </Grid>
</Window>
```

### В.7 C# ПРОХОДЖЕННЯ ТЕСТУ

```
using Education_Tests_System.models;
using System;
using System.ComponentModel;
using System.Linq;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Threading;

namespace Education_Tests_System.views
{
  /// <summary>
  /// Логика взаимодействия для TakeSurvey.xaml
  /// </summary>
  public partial class TakeSurvey : Window, INotifyPropertyChanged
  {
    private User user;
    private Test test;
    private TestSurvey survey; // null for offline mode
```

```
    private DispatcherTimer timer;

    //todo duration limit!!!
    public int TakenQuestionCount
    {
        get
        {
            int count = 0;
            foreach (var question in test.Questions)
                foreach (var answer in question.Answers)
                    if (answer.IsUserSelected)
                    {
                        count++;
                        break;
                    }
            return count;
        }
    }

    public event PropertyChangedEventHandler PropertyChanged;
    public void OnPropertyChanged(string prop) => PropertyChanged?.Invoke(this, new
PropertyChangedEventArgs(prop));

    public TakeSurvey(Test test)
    {
        InitializeComponent();

        this.test = test;
        BtnCheck.DataContext = test;
        PrepareControls(test);
    }
    public TakeSurvey(User user, Test test)
    {
        InitializeComponent();

        this.user = user;
        this.test = test;
        BtnCheck.DataContext = test;
        survey = new TestSurvey() { IdEntity = user.IdEntity, IdTest = test.Id };
        PrepareControls(test);
    }

    private void PrepareControls(Test test)
    {
        Title = test.Name;
```

```
    QuestionReadyCount.DataContext = this;
    foreach (var question in test.Questions)
    {
        TabItem questionTab = new TabItem
        {
            Header = "Питання " + (TestSurveyTab.Items.Count + 1),
            ContentTemplate = (DataTemplate)FindResource("QuestionTab"),
            DataContext = question,
        };
        Binding binding = new Binding();
        binding.RelativeSource = RelativeSource.Self;
        binding.Path = new PropertyPath("DataContext");
        questionTab.SetBinding(ContentProperty, binding);
        TestSurveyTab.Items.Add(questionTab);
    }
    if (test.Duration == null)
        TimePanel.Visibility = Visibility.Hidden;
    if (survey != null)
    {
        survey.DateBegin = DateTime.Now;
        if (test.Duration != null)
        {
            timer = new DispatcherTimer();
            timer.Interval = new TimeSpan(0, 0, 1);
            timer.Tick += new EventHandler(CheckUserTime);
            timer.Start();
        }
    }
}
private void CheckUserTime(object sender, EventArgs e)
{
    DateTime now = DateTime.Now;
    TimeSpan delta = now - (DateTime)survey.DateBegin;
    TimeLeft.Content = ((TimeSpan)(test.Duration - delta)).ToString(@"hh\:mm\:s");
    if (delta >= test.Duration)
    {
        survey.DateEnd = now;
        timer.Stop();
        ProceedTestResult();
    }
}

private void Window_Closed(object sender, EventArgs e) => Owner?.Show();
```

```
    private void TestSurveyTab_SelectionChanged(object sender, SelectionChangedEventArgs e)
    {
        OnPropertyChanged(nameof(TakenQuestionCount));
    }

    private void Previous_Click(object sender, RoutedEventArgs e)
    {
        OnPropertyChanged(nameof(TakenQuestionCount));
        if (TestSurveyTab.SelectedIndex != 0)
            TestSurveyTab.SelectedIndex--;
    }

    private void Next_Click(object sender, RoutedEventArgs e)
    {
        OnPropertyChanged(nameof(TakenQuestionCount));
        if (TestSurveyTab.SelectedIndex != TestSurveyTab.Items.Count - 1)
        {
            TestSurveyTab.SelectedIndex++;
            return;
        }
        // check not taken question
        foreach (TabItem tab in TestSurveyTab.Items)
        {
            Question question = tab.DataContext as Question;
            int selectedAnswerCount = question.Answers.Where(item => item.IsUserSelected).Count();
            if (selectedAnswerCount == 0)
            {
                TestSurveyTab.SelectedItem = tab;
                MessageBox.Show("Оберіть відповідь на дане питання", "Любий користувач", MessageBoxButton.OK, MessageBoxImage.Warning);
                return;
            }
        }

        if (survey != null)
            survey.DateEnd = DateTime.Now;
        ProceedTestResult();
    }

    private void ProceedTestResult()
    {
        float max_points = 0, user_points = 0, user_points_per_question, points_per_answer;
```

```
        int right_user_answer_count, wrong_user_answer_count;
        foreach (var question in test.Questions)
        {
            user_points_per_question = 0;
            right_user_answer_count = 0;
            wrong_user_answer_count = 0;
            max_points += question.Points;
            points_per_answer = question.Points / question.RightAnswersCount;
            foreach (var answer in question.Answers)
            {
                if (answer.IsUserSelected)
                {
                    if (survey != null)
                    {
                        survey.Answers.Loaded = true;
                        survey.Answers.Add(new AnswerSurvey() { IdQuestion = question.Id,
IdAnswer = answer.Id });
                    }
                    if (answer.IsRight)
                        right_user_answer_count++;
                    else
                        wrong_user_answer_count++;
                }
            }
            user_points_per_question = (question.ChooseAllAnswers ?? false) ?
                                    points_per_answer * (right_user_answer_count -
wrong_user_answer_count):
                                    points_per_answer * right_user_answer_count;
            user_points += user_points_per_question > 0 ? user_points_per_question : 0;
        }


        if (survey != null)
        {
            try
            {
                DbHelper.BeginTransaction();
                survey.Points = user_points;
                survey.Save();
                foreach (var answerSurvey in survey.Answers)
                {
                    answerSurvey.IdTestSurvey = survey.Id;
                    answerSurvey.Save();
                }
                DbHelper.CommitTransaction();
```

```
            }
            catch
            {
                DbHelper.RollbackTransaction();
                MessageBox.Show("Вибачте, під час збереження даних до бази виникла помилка. Перепройдіть тест, будь ласка. \nСистема запам'ятала Ваші відповіді", "Увага", MessageBoxButton.OK, MessageBoxImage.Error);
                Close();
                return;
            }
        }

        MessageBox.Show("Ваш результат: " + user_points + " з " + max_points, "Любий користувач", MessageBoxButton.OK, MessageBoxImage.Information);
        if (test.IdKind == TestKind.Trainy)
            foreach (var question in test.Questions)
                foreach (var answer in question.Answers)
                    if (answer.IsUserSelected)
                        answer.IsUserSelected = false;
        Close();
    }

    private void Answer_Click(object sender, RoutedEventArgs e)
    {
        Question question = (TestSurveyTab.SelectedItem as TabItem).DataContext as Question;
        int selectedAnswerCount = question.Answers.Where(item => item.IsUserSelected).Count();
        int answerId = Convert.ToInt32((sender as CheckBox).Tag);
        // prevent from selecting odd answers if there is no 'choose all' option
        if (!(question.ChooseAllAnswers ?? false) && selectedAnswerCount > question.RightAnswersCount)
            question.Answers.Single(item => item.Id == answerId).IsUserSelected = false;
    }

    private void Check_Click(object sender, RoutedEventArgs e)
    {
        TabItem tab = TestSurveyTab.SelectedItem as TabItem;
        Question question = tab.DataContext as Question;
        int selectedAnswerCount = question.Answers.Where(item => item.IsUserSelected).Count();
        if (selectedAnswerCount == 0)
        {
            MessageBox.Show("Оберіть відповідь на дане питання", "Любий користувач", MessageBoxButton.OK, MessageBoxImage.Warning);
```

```
                return;
            }
            question.IsUserFinished = true;
        }
    }
}
```